NASA CONTRACTOR REPORT

NASA CR-1

NASA CR-1127

PRACTICAL RELIABILITY

Volume II - Computation

Prepared by
RESEARCH TRIANGLE INSTITUTE
Research Triangle Park, N. C.
for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • AUGUST 1968

NASA CR-1127

PRACTICAL RELIABILITY

Volume II - Computation

Prepared under Contract No. NASw-1448 by
RESEARCH TRIANGLE INSTITUTE
Research Triangle Park, N.C.

for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

For sale by the Clearinghouse for Federal Scientific and Technical Information
Springfield, Virginia 22151 – CFSTI price $3.00

FOREWORD

The typical few-of-a-kind nature of NASA systems has made reliability a premium even on the initial items delivered in a program. Reliability defined and treated on the basis of percentage of items operating successfully has much less meaning than when larger sample sizes are available as in military and commerical products. Reliability thus becomes based more on engineering confidence that the item will work as intended. The key to reliability is thus good engineering--designing reliability into the system and engineering to prevent degradation of the designed-in reliability from fabrication, testing and operation.

The PRACTICAL RELIABILITY series of reports is addressed to the typical engineer to aid his comprehension of practical problems in engineering for reliability. In these reports the intent is to present fundamental concepts on a particular subject in an interesting, mainly narrative form and make the reader aware of practical problems in applying them. There is little emphasis on describing procedures and how to implement them. Thus there is liberal use of references for both background theory and cookbook procedures. The present coverage is limited to five subject areas:

Vol. I. - Parameter Variation Analysis describes the techniques for treating the effect of system parameters on performance, reliability, and other figures-of-merit.

Vol. II. - Computation considers the digital computer and where and how it can be used to aid various reliability tasks.

Vol. III. - Testing describes the basic approaches to testing and emphasizes the practical considerations and the applications to reliability.

Vol. IV. - Prediction presents mathematical methods and analysis approaches for reliability prediction and includes some methods not generally covered in texts and handbooks.

Vol. V. - Parts reviews the processes and procedures required to obtain and apply parts which will perform their functions adequately.

These reports were prepared by the Research Triangle Institute, Research Triangle Park, North Carolina 27709 under NASA Contract NASw-1448. The contract was administered under the technical direction of the Office of Reliability and Quality Assurance, NASA Headquarters, Washington, D. C. 20546 with Dr. John E. Condon, Director, as technical contract monitor. The contract effort was performed jointly by personnel from both the Statistics Research and the Engineering and Environmental Sciences Divisions. Dr. R. M. Burger was technical director with W. S. Thompson serving as project leader.

This report is Vol. II - Computation. It serves in a support role to the other volumes, particularly to Vol. I - Parameter Variation Analysis and Vol. IV - Prediction, by treating the computer techniques for implementing the reliability tasks developed in the other volumes. R. L. Beadles is the principal author of this report. A. C. Nelson made major contributions to Secs. 2 and 8; he and J. R. Batts wrote the computer programs discussed in Sec. 7.

ABSTRACT

This report places in perspective the role of automatic digital computations in design for reliability. It is intended for the design engineer, the systems engineer, and the test engineer as well as the reliability specialist. The degree of detail with which the various topics are treated is sufficient to enable the engineer not previously familiar with the subject to properly select and use the methods presented.

As a fundamental introduction to automatic digital computation, the report first briefly describes the computer, how it is used, and some of the mathematical problem types that are amenable to computer solution. The orientation to reliability is then provided in a brief perspective of reliability tasks and the relation of the computer to them. Later sections of the report treat specific reliability tasks and explore the mathematical methods related to them and how the computer is used to implement them. Some specific computer programs are identified and their uses illustrated by examples. Parameter variation analysis and reliability prediction are treated in more detail than others since these areas of application are particularly suited to computer methods. The last section of the report summarily treats some recent developments in communicating with the computer which make it more suitable to engineering and reliability applications.

TABLE OF CONTENTS

		Page
	FOREWORD	iii
	ABSTRACT	v
1.	Introduction	1
2.	Fundamentals of Digital Computation	3
	2.1 Digital Computer Concepts	3
	2.2 Computer Programming Languages	7
	2.3 Basic Mathematical Problems that Can Be Solved by a Computer	9
3.	Reliability and the Computer--A Perspective	23
4.	Parameter Variation Analysis	28
	4.1 PVA Modeling	28
	4.2 Analysis Techniques	30
	4.2.1 Worst-Case Analysis	30
	4.2.2 Sensitivity Analysis	33
	4.2.3 Moments Analysis	34
	4.2.4 The Convolution Method	36
	4.2.5 Monte Carlo Analysis	37
	4.3 PVA Computer Programs	40
	4.3.1 A General PVA Program	42
	4.3.1.1 General PVA Program Example	49
	4.3.2 ECAP and NASAP for PVA	58
5.	Part Application Analysis	61
6.	Failure Modes and Effects Analysis (FMEA)	62
7.	Reliability Prediction	66
	7.1 Developing the Prediction Model	67
	7.2 Making the Reliability Prediction	68
	7.3 Reliability Prediction Programs	70
	7.3.1 A Computer Program for System Reliability	70
	7.3.2 Reliability Cost Trade-Off Analysis Program	84
8.	Testing	94
	8.1 Attribute Data	94
	8.2 Variables Data	98
	8.2.1 Failure-Time Data	98
	8.2.2 Performance Measurements at Discrete Time(s)	99
	8.2.3 Continuous Recording of Performance Measurements	101
	8.3 Stress-Strength Measurements	102

TABLE OF CONTENTS (CONT'D)

		Page
9.	Trends in Digital Computation	105
	Appendix A	109
	Appendix B	119
	Appendix C	125

1. Introduction

The digital computer has had a significant impact on engineering design and development. Because of it, larger and more sophisticated systems have become realities rather than mere dreams. But with these developments, the achievement of system reliability has become more difficult. The designer's task of building in the reliability is a complex one involving extensive analysis and computation, and it is only natural that the computer be employed to its full capacity here also.

A good, reliable design results from a continual assessment and improvement process. Performance analysis, testing, failure mode and effects analysis, and reliability prediction are typical, key tasks in this iterative process. As a tool of the designer, the computer must contribute directly to performance of such tasks.

The purpose of this report is to place in proper perspective the role of automatic digital computations in design for reliability. It is intended for the design engineer, the systems engineer, and the test engineer as well as the reliability specialist. The degree of detail with which the various topics are treated is sufficient for enabling the engineer not previously familiar with the subject to properly select and use the methods presented.

Of equal importance to an appreciation for what the digital computer can do is an adequate appreciation for what it cannot do. Consequently, care is taken at appropriate points to indicate the limitations of the available computer methods and programs.

As a fundamental introduction to automatic digital computation, Sec. 2 briefly describes the computer, how it is used, and some of the mathematical problem types that are so common in many uses of the computer. The orientation specifically to design reliability applications is provided in Sec. 3 which gives a brief overall perspective of the engineering tasks and relates the role of the computer to them.

Secs. 4 through 8 separately treat specific design tasks and explore in more depth the mathematical methods and how the computer is used to implement them. Some specific computer programs are identified and their uses illustrated by examples. Parameter variation analysis and reliability prediction are treated in more detail than others since these areas of application are particularly suited to computer methods. Sec. 9 briefly summarizes the state-of-the-art in automatic digital computation emphasizing those recent developments in communicating with the computer which make it more suitable to engineering application.

The computer output can be no better than the model used to obtain it. Before a computer program can be written to analyze a piece of equipment, a conceptual model of that piece of equipment must be formulated. Before existing computer programs can

be used intelligently, the models they assume and the relationships of those models to the equipment which is to be analyzed must be known. Of particular importance is the knowledge of the parameter ranges over which the models assumed by a computer program are valid and how these ranges relate to a valid model for the equipment to be analyzed. A good discussion on the practical aspects of modeling is presented in Sec. 2.1, Vol. I - Parameter Variation Analysis of this report series.

2. Fundamentals of Digital Computation

The purpose of this section of the report is to treat in as brief a manner as is consistent with clarity the fundamentals of the digital computer and its use.

2.1 Digital Computer Concepts

A digital computer system is comprised of two elements which have come to be called hardware and software. The hardware consists of the physical pieces of equipment, viz, the central processor, the card and tape readers, the information storage media, and the printers and plotters. The software consists of all the computer programs which are available to cause the various pieces of equipment to do useful things.

A simplified block diagram of a stored-program electronic digital computer is shown in Fig. 2-1. The organizational structure shown in the figure is common to every modern digital computer although some computers may have more than one memory unit, arithmetic unit, etc. Although digital computers other than stored-program electronic digital computers are of historical interest they are not of interest in modern engineering. In this report when we use the word computer we shall mean stored-program electronic digital computer.

The function of a computer is to take data via the input unit from the external world, perform calculations on it as specified by the program stored in the memory unit, and supply the results via the output unit to the external world. In a typical installation the input unit is a punched card reader which reads the information on the cards into the memory unit under control of the control unit. The typical output unit is the line printer, which produces a printed copy of the results of the calculations.

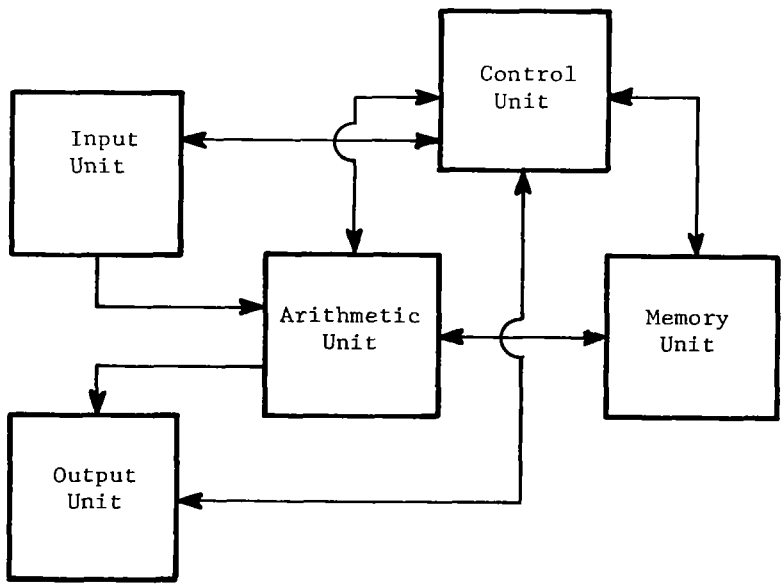


Figure 2-1. Basic Computer Organization

Computers are widely used both in real-time operation and in off-line operation. Although the terms real-time and off-line are relative to the application, the meaningful distinction usually is that in the real-time application, the input data must be processed rapidly and an output produced so that some kind of response can be quickly initiated. An example of the real-time application of the digital computer is in conjunction with a radar installation. There the input data comes from the radar and must be processed sufficiently rapidly to compute, for example, guidance commands for a missile launched to intercept an attacking aircraft. We will not discuss in this report the use of digital computers in such real-time applications.

Referring again to Fig. 2-1 we consider briefly the function of each of the blocks shown. First, the memory unit serves as storage for (1) the program which is to be executed, (2) the input data until it is needed for processing, (3) intermediate results during the execution of the program, and (4) the final results until they are ready for output. The memory unit typically is a principal element of the computer; the cost and speed of the modern digital computer are largely governed by the cost and speed of the memory. It is not uncommon for the cost of the memory to approach the cost of all the other units combined.

The memory contents are stored in the form of binary digits (bits) which are grouped into blocks of sufficient size for the number range and precision requirements for which the computer is designed. Such a block of binary digits is called a memory word. In computers in common use today the memory word varies from 12 bits up to 60 bits, which corresponds to a decimal number range of 4000 to 10^{18}. The number of words that a computer memory may store also varies widely and ranges from 1000 words up to 10^6 or more words.

Associated with the memory unit are two registers*. These are the memory address register and the memory data register. When it is desired to store a number in memory or retrieve it from memory, it is necessary to give the location of the particular memory word desired. The memory address register is used to designate the address, i.e. the location, of the word in memory. When the command is given by the control unit to store or retrieve a word from memory, the memory address register is used to designate the address. There are as many unique addresses, i. e. locations at which a number can be stored, in the memory as the number of words which the memory is capable of storing.

The memory data register is used as an intermediate storage when a word is going from the arithmetic unit or the input unit to the memory. To store a word in memory,

---

*A register is a temporary storage device. It typically can store one memory word.

the word is placed into the memory data register, and the address at which it is to be stored is placed into the memory address register. Then the store command generated by the control unit causes the word to be stored at the specified address. When a data word is to be retrieved from memory, the address of the word is again placed into the memory address register, and the fetch command from the control unit causes the word to be transferred from the specified address in the memory to the memory data register.

The arithmetic unit performs an arithmetic (or logic) operation as specified by the program between a word contained in a register in the arithmetic unit called the accumulator and a word fetched from memory into the memory data register. This description holds for the single address computer. The term single address means simply that a single program step (which also is stored as a word in memory but is called an instruction word) specifies the address of only one data word in memory. The second word to be used in an operation is contained in the accumulator register in the arithmetic unit. Althought some computers specify more than one address in one instruction word, the single address computer organization is the most widely used.

In the single address computer, the accumulator register contains one operand* for an operation, with the other operand being first in memory and later in the memory data register. The result of an operation usually ends up in the accumulator. Data words can be fetched from memory to the accumulator or stored from the accumulator into the memory. Except when the computer instruction specifically calls for it, the contents of the accumulator are not disturbed by an operation.

The control unit is the logic complex which determines which operation is to be performed at what time and what sequence of elementary logic steps accomplishes the operation. The control unit contains two very important registers--the program register (also called the instruction counter) and the instruction register.

The program stored in the computer memory unit consists of a sequence of instructions which the computer is to perform. The program is stored in the memory in the proper sequence: the first instruction is stored in some location n, the second stored in location n+1, etc. The function of the program register is to keep track of the location from which the next instruction is to be fetched; it does this by counting the instructions as they are performed. Unless specifically requested to do so by a specific instruction, the program will proceed in sequence by picking up its instructions from successive memory addresses.

---

*An operand is any single-word quantity which is operated upon by the computer.

The function of the instruction register is to temporarily store each instruction to enable the control unit to decode it and initiate and properly time the sequence of elementary logic steps which implements the instruction. It is a fundamental fact that the memory contains both the instructions tobe executed, (i.e. the program) and the data on which the instructions are to operate.

The two kinds of stored words (instructions and data) are treated in two entirely different ways. An instruction is transferred to the instruction register where it is examined by the control unit to determine:

(1) what operation (add, subtract, logic, etc.) is required,

(2) where the second operand is located, i.e., the address of the second operand, and

(3) where the result of the operation should be placed.

If, as is usually the case, one of the operands is contained in the memory, then this operand address is contained in the instruction word located in the instruction register. This address is furnished to the memory address register at the correct time as specified by the control unit. The result of the operation usually goes into the accumulator.

The input unit and output unit have associated with them a data register and an address register analogous to the memory address register and memory data register of the memory unit. Data coming from an external device is placed into the input-output (I/0)data register and later transferred into the accumulator for use inside the computer. Data going to an external device is transferred from the accumulator to the I/0 data register from which it is removed by the I/0 device. Since typically several input-output devices are connected to the computer, input-output addresses must be specified to identify which I/0 device is requested. The function of the I/0 address register is to designate the address of the I/0 device; the address of the I/0 device is nothing more than a number which it has been given to uniquely identify it.

A computer can perform only the operations which have been built into it. The list of operations which a computer can perform is called the instruction repertoiré of the computer. Any program which can be executed by a computer is made up of only those instructions contained in that computer's instruction repertoiré.

An instruction is a step in a program but we wish to indicate in detail what comprises an instruction. For purposes of discussion the following description of an instruction is referenced to a single address computer. A computer instruction is made up of three basic parts:

The operation code (op code) is that part of an instruction which specifies to the control unit which operation is to be performed (add, subtract, transfer data to or from memory, etc.)

The instruction modifier is a group of bits which further specifies how the instruction is to be performed. For example, the add operation ordinarily results in the sum being placed in the accumulator only. A modifier to the add op code might specify that the result of the add operation also be placed into a memory location.

The address of the second operand is the third basic part of an instruction. The address is simply the number of the memory location which contains the data to be operated on as specified by the op code.

The computer has two characteristics which make it an exceedingly powerful aid to problem solving. First, the computer can perform operations (albeit simple) exceedingly rapidly. It is not uncommon for a large-scale modern computer to be able to perform, for example, one million addition operations in one second*. Fundamental to the ability to do simple operations exceedingly fast is the ability to obtain the data at an adequately rapid rate. The availability of the data at such a rate implies that both the instructions for operating on the data and the data itself must be stored in the computer memory.

The second characteristic of fundamental importance in the computer is its ability to perform the same sequence of operations an arbitrary number of times, except that the sequence is performed each time on a different set of data--this is the ability of the computer to modify its own program. It might appear that to instruct a computer to perform the operations necessary to add by pairs two tables of 100 numbers each would require 200 or more instructions. On the contrary, it is a simple matter to put instructions in the program which modify the instruction addresses in a way to step through the pairs of numbers in the tables and make the total number of required instructions something like ten.

2.2 Computer Programming Languages

In the final analysis a digital computer can only recognize binary patterns. Thus there are several programs between the computer programmer using FORTRAN (or another high-level programming language) and the actual execution by the computer of the operations requested by the programmer in his FORTRAN program. Three levels of computer languages are in wide use today: assembly languages, procedure-oriented languages such as FORTRAN, and problem-oriented languages such as the input language for automatic circuit analysis programs.

---

*The response time of the logic devices internal to a modern computer is a few nanoseconds, which is an interesting contrast to the few milliseconds response time of the neurons of the computer user.

The assembly language* is the first level of computer language removed from the binary patterns which the computer directly recognizes. Consider the add operation. The binary pattern for the add operation(which is the add op code)for a particular computer might be 1000. Before the computer can actually execute an add operation, it must have in the op code portion of its instruction register the binary pattern 1000. It also must have, in the address field of the instruction word, the binary pattern which gives the location of the memory word containing the data which is to be added to the contents of the accumulator. The assembly language enables the programmer to use a suggestive sequence of letters called an instruction mnemonic, for example ADD in the case of the add operation, to specify that an addition is to be performed. Before this addition operation called for by the assembly language program can be performed in the computer, it must be processed by another computer program--called the assembler--which has the ability to interpret the letters ADD as the op code 1000 for the add operation. If we wish to add the numbers X and Y, the availability of the assembly language enables us to write a sequence of instructions which loads the accumulator with X, adds Y, and stores the result at a desired location Z. Such a sequence is

```
LDA     X
ADD     Y
STO     Z
```

where LDA, ADD, and STO are respectively the mnemonics for loading the accumulator from the memory, adding to the accumulator, and storing the contents of the accumulator in the memory. Each of the letters X, Y, and Z represents the symbolic address of a memory word. The assembler in addition to converting the instruction mnemonics to their binary equivalents, allocates memory words and converts each symbolic address used in an assembly language program to a fixed binary memory address. Thus assembly language programming contrasts to having to write the binary patterns for each computer instruction and to allocate memory locations by writing a binary memory address for each data word used in the program.

The procedure-oriented language, of which FORTRAN** is the best known and most widely used example, effectively removes the programmer one level further from the tedious task of programming the computer with binary patterns. Thus, whereas three assembly language instructions were required to specify the addition of X and Y and

---

*Assembly language is also called machine language, since the details of an assembly language are highly dependent on the details of the specific machine(the specific computer) on which it is used. Originally, machine language meant the binary patterns directly recognized by a computer.

**FORTRAN is a contraction of "formula translation".

store the result in the memory at location Z, the FORTRAN statement for accomplishing this would be simply Z = X + Y. The program which processes the FORTRAN statement (called the FORTRAN compiler) would produce the same sequence of binary patterns that the assembly language instructions produce. Whereas in writing in assembly language one statement must be written for each instruction to be executed, a FORTRAN statement (and in general any procedure-oriented language statement) will produce several computer instructions, typically four or five.

An advantage of procedure-oriented languages which is probably more important than their ease of use by the programmer is that a procedure-oriented language program is nearly machine independent, in dramatic contrast to the program written in assembly language. Thus a program which is written in FORTRAN can be interpreted, via the FORTRAN compiler of any computer which has one, and then executed on that computer, with only minor program changes between different computers. A specific computer almost never stays in a particular installation for more than a few years. The use of procedure-oriented language programming is the only effective way to prevent losing the large investment in programming time and checked-out programs for the old computer when the new computer is installed.

The problem-oriented language is the newest and in many ways the most powerful computer language. A single statement in a problem-oriented language might result in the execution of up to several thousand computer instructions. Problem-oriented languages are discussed in later sections of the report. In essence they consist of the input languages to special programs written to aid in specific problem areas, e.g., problems in network analysis.

In the final analysis, the computer can do no more and no less than precisely what it is instructed to do via the program. Given adequately clever people preparing and using the computer programs, the computer can indeed do some very impressive things. As an aid to design for reliability, the computer enables equipment designers to conduct many more and more thorough analyses of their designs than would be possible by any combination of hand calculation and laboratory experimentation. However, it is up to the computer users to examine the output from the programs they are using, to interpret the computer results, and themselves to make the corrections and design modifications which they discover via computer analysis. The computer does not by any stretch of the imagination remove the need for good engineering and clear thinking in the development and design of reliable equipment.

2.3 Basic Mathematical Problems that Can Be Solved by a Computer

Problem solving is an essential part of engineering design. Some of the problems are very simple from a computational standpoint, requiring only a slide rule, a pencil, and a piece of paper, while other problems require a team of engineers working

many days or perhaps years. The latter problems were attacked by approximations based on simplifying assumptions when digital computers were not available. However, it is now practical to evaluate the adequacy of such assumptions and delve into system analysis problems which would have been impractical only a few years ago.

Solving a particular engineering problem on a computer usually requires the use of several basic mathematical techniques. For example, suppose that we wish to obtain the minimum value of a particular known function f(x) on a certain interval [a,b]. In some cases the derivative function can be written without difficulty, the resulting equation solved for the zeros, and the solutions tested to determine which value of the independent variable yields the minimum value of the response or performance variable. However, in some problems the writing of the derivative takes considerable time and its evaluation a great deal longer time than the evaluation of the original function, and often the equation obtained by equating the derivative to zero is hard to solve. Hence a computer is used to aid in the analysis.

Again there are many avenues of attack on the problem. One approach is to evaluate the function f(x) at a single value of x within the given interval and then select another x value at some predetermined distance from the first point and compare the two values. If the value of the function at the second point is less than at the first point, take it as a new reference point and proceed to a third point, etc. In such a process the interval of step size between successive x's must be decreased in a systematic manner when no improvement results from increasing or decreasing x by the prescribed step size. Such a procedure will ultimately lead to an adequate solution of a problem of a local minimum, and in the case of a convex function* on the interval an absolute minimum, as seen in the figure below.

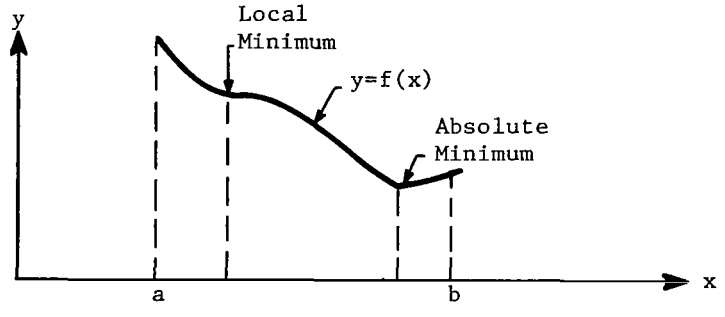


Figure 2-2. Minima of a Function f(x)

*Linear interpolation never underestimates the real value of a convex function at the interpolated point. For a mathematical definition of a convex function, see Ref. 2.2.

Another attack on the problem is to select three points on the interval [a,b], fit the corresponding y's by parabola, and estimate the location of the vertex. Then, select three new points in the neighborhood of this vertex and repeat the above; eventually the location of the local minimum point is determined to within the desired degree of precision. This approach requires the evaluation of the function at three points and the solution of a set of three linear simultaneous equations for each iteration. It also requires the provision of a logical procedure for altering the step size as the iterations converge toward the solution.

If the function is convex and only one independent variable is involved, there is a near optimum procedure for finding the minimum using the properties of the Fibonacci numbers 1, 1, 2, 3, 5, 8, 13, 21, ..., where each number in the sequence is obtained by adding the two previous numbers, that is

$$a_{n+1} = a_n + a_{n-1} \cdot \qquad (2\text{-}1)$$

This procedure is considered in Ref. 2-2 under the basic problems of optimization. Also see Ref. 2-3 for a mathematical treatment of this subject.

In the following sections are presented some of the basic problem types, some of the approaches to solution, and the relation of these basic problems to typical engineering problems via particular computer programs. This approach was selected to avoid some of the redundancy which would occur as a result of treating problems in electronics, or propulsion, or structures as separate problems when in fact they may be all of the same basic problem area.

Function Evaluation

The first problem type is one of evaluating a function of one or several variables defined by

$$y = f(x_1, x_2, \ldots, x_n)$$
$$= f(\underline{x})$$

where $\underline{x} = (x_1, \ldots, x_n)$ and x_i is the i-th variable. For simple functions a computer is not needed to solve for y for given values of the x_i; however, if the operation is to be repeated frequently or if the function is complex the use of a computer is often justified. The display of the output in a table or graph form is important from the user standpoint. If the function is an important one a table of values for future use can be prepared for different values of $\underline{x}$. It is obvious that a computer can be used to obtain reams of paper containing numerical values of y for various combinations of x_i, i=1, ..., n. However, the objective of the problem and the uses to be made of the results should be thoroughly considered prior to computation. There is no

need to tabulate a function which can be computed almost as readily by hand as one can locate the table and then look it up. Although this statement seems obvious it is possible to locate examples of such functions tabulated in the literature. Also, the selection of the values of the x_i, i=1, ..., n at which to compute the y's is an important aspect of the problem.

In engineering applications the performance or some figure-of-merit (FOM) of an equipment can often be expressed as a function of the characteristics of its parts and the inputs, environments, loads, etc. Thus the FOM may be obtained for various values of the variables which influence it. Computation of static and dynamic responses with circuit and structural equations are typical examples in engineering.

Functional Equation

Next consider the inverse problem of solving for $\underline{x}$ given y, i. e., if

$$y_o = f(\underline{x})$$

determine $\underline{x}$ such that $f(\underline{x}) = y_o$, where the solution(s) will be denoted by $\underline{x}_o$. For example, we may have an algebraic equation in one variable x and wish to solve for the values of x at which the curve corresponding to the equation y = f(x) crosses or intersects the x axis (line $y \equiv 0$). We may wish to obtain the extreme points (maxima, minima, points of zero derivative) for f(x) when the derivative function f'(x) can be readily obtained. In general the problem may require the use of an iteration technique, such as the Newton-Raphson method* of solving an equation by using the construction of successive tangents to the curve at points approaching the solution.

A typical engineering example of the above problem is to find the parameter values yielding a given level of performance. It is possible to obtain contours of equal performance values of the set of all values of the independent variables corresponding to $y = y_o, y_1, \ldots, y_m$. Such a set of contours is indicated in Fig. 2-3. Such techniques can be helpful in determining the operating conditions yielding the desired performance. The above technique becomes very helpful when two or more dependent or performance variables are being considered. For example, in Fig. 2-4 two variables are shown and the region of operation defined by the set of the x_i, i=1, 2, for which $y_1 \geq 30$, $y_2 \leq 20$. The shaded region provides a region of operation which satisfies the given constraints. Further discussion of such an approach and practical problems associated with it are in Vol. I - Parameter Variation Analysis of this series.

---

*There are numerous texts on standard numerical methods. Refs.2-4 and 2-5 are good starting points.

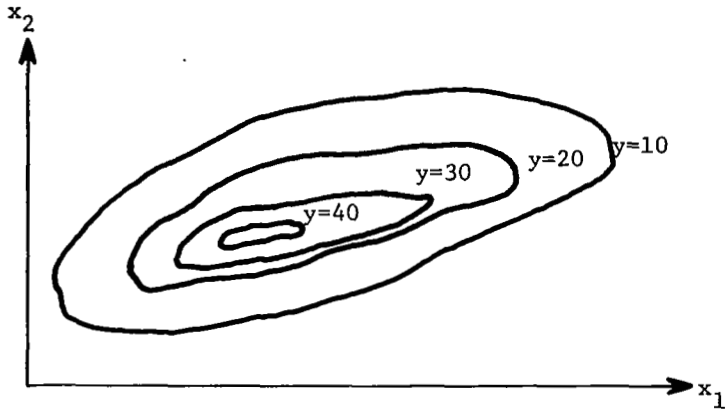


Figure 2-3. Typical Performance Contours

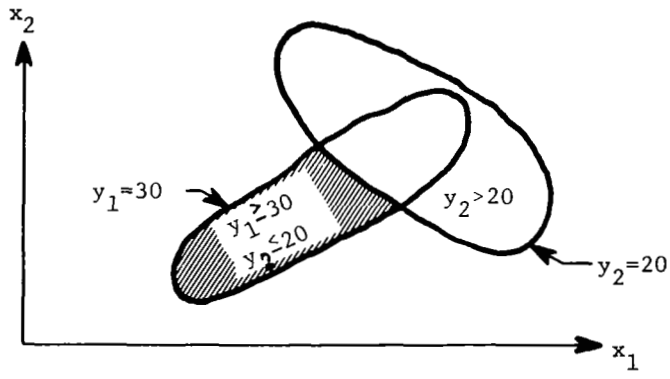


Figure 2-4. Region of Desired Performance

If a system of equations is involved the problem may have a single solution or a multiple solution depending on the degree of the equations, the number of equations relative to the number of unknowns, etc. Many problems in real world applications result in a system of equations to be solved for the value or values of the unknown variables which satisfy specified conditions. Some of these problems will be considered later.

Functional Approximation

Another important problem in computer application is the use of functional approximations to functions which cannot be expressed in a closed form,e. g., some indefinite integrals or the sum of an infinite series. For example, the approximations to sin x and e^x can be in the form of a Taylor series or orthogonal polynomials such as Chebyshev, Legendre, and Hermitian polynomials. In many applications a finite Taylor series approximation is to be used. On the other hand, extremely accurate approximations are sometimes needed, such as for the cumulative probability integral of the Gaussian distribution. Rational integral functions are often used in approximating such curves. See Ref.2-5 for examples of approximations to a variety of functions.

One useful application in engineering problems is reducing a complex function to a linear or , when necessary, to a second degree approximation. Such an approach is useful in deriving the properties of the distribution of a performance variable y in terms of the characteristics of the distributions of the independent variables. It is also applied often in constructing contours and performing sensitivity analyses. A linear approximation is most often sufficient over the region of interest.

This problem type leads logically into the problem area of curve fitting which is discussed below. The two problems are separated here because the first problem type deals with a known model defined explicitly such as

$$y = e^x$$

or

$$y = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\ \sigma_u} \exp\left(\frac{u}{\sigma(u)}\right)^2 du$$

or only implicitly such as

$$f(t,\ y,\ dy/dt,\ldots) = 0.$$

The curve fitting problem on the other hand treats a given model form with unknown constants to be determined from given data.

Curve Fitting

Suppose that instead of being given a function as suggested above one is given a set of values y_i and the corresponding x_i or $\underline{x}_i$ in the case of several independent variables. For one independent variable and one dependent variable, a curve may be fitted to the data freehand. If however we have some knowledge concerning the underlying mechanism (a model form) and wish to estimate certain constants or parameters of the model, a more appropriate procedure would be to estimate the parameters by a mathematical procedure such as the method of least squares. Even when the model form is not known, there is often considerable advantage in fitting the curve by a mathematical interpolation or a graduation formula such as a linear or second degree function in x or possibly in 1/x depending on the nature of the given data. Such a prediction equation is satisfactory only in the region of the given data unless theoretical knowledge is available to allow correct extrapolation beyond the region of experimental results given by the data.

Another closely related technique for fitting a curve is smoothing the data. Smoothing the data is based on the fitting of polynomials to a set of successive data points and calculating the "smoothed" points. For example, suppose that $2t + 1$ successive equally-spaced points, ($t = 1,2,\ldots$) are selected and a polynomial of degree three fitted to these points. Then the smoothed value of y is given by

$$y_2^* = \frac{1}{35}\,(-3y_o + 12y_1 + 17y_2 + 12y_3 - 3y_4) \qquad (2\text{-}2)$$

where $y_o, y_1, \ldots, y_4$ are five consecutive values of y.

The least squares technique has the most useful application when fitting a curve to a set of observed (experimental) data points. Suppose that one hypothesizes that the mean value of the performance variable y of given $\underline{x}$ is a linear function of certain functions $f_i(\underline{x})$ of the independent variables x_i, i=1, ...,n. The expected value of y is

$$E\{y|\underline{x}\} = \beta_o + \sum_{i=1}^{p} \beta_i\, f_i(\underline{x}). \qquad (2\text{-}3)$$

or

$$\eta = \beta_o + \Sigma\beta_i\, f_i(\underline{x}),$$

where η denotes the mean value of y for given values of $\underline{x}$. For example, if $f_i(\underline{x}) = x_i$ and $p = n$, then

$$\eta = \beta_o + \beta_1 x_1, \ldots + \beta_n x_n \qquad (2\text{-}4)$$

is a linear function of the x_i. If one the other hand $f_i(\underline{x}) = 1/x_i$ and $p = n$, then

$$\eta = \beta_o + \beta_1/x_1 + \ldots + \beta_n/x_n \tag{2-5}$$

is not a linear function of the x_i's. However, it is a linear function of the β's which are to be estimated from the given data by the method of least squares. Thus the estimates $b_o, b_1, \ldots, b_p$ of $\beta_1, \ldots, \beta_p$ are given by the values of the β's which minimize the sum of square of deviations

$$s = \Sigma\{y - \beta_o - \Sigma\beta_i f_i(\underline{x})\}^2. \tag{2-6}$$

Certain assumptions are made in this solution, namely that the $y_i = f_i(\underline{x})$ are distributed about the corresponding means $\eta_i = \beta_o + \Sigma\beta_i f_i(\underline{x})$ with constant variance and that they are independent observations. The solution to the least squares problem is obtained by solving a set of $p + 1$ equations in $p + 1$ unknowns, often referred to as normal equations in the literature [Ref. 2-4].

In many physical problems the model form cannot be expressed as simply as above (i.e. as a linear function in the unknown constants β_i, i=0, 1, ..., p), but is nonlinear, such as

$$y = \beta_o(1 - e^{-\beta_1 x}).$$

In such examples iterative procedures must be used to solve for the best estimates of the constants in the least squares sense. For example, see Ref. 2-6 concerning two basic approaches. Computer programs have been written to perform the iteration. See Ref. 2-7 for example. This problem requires the use of a general technique for solving a system of nonlinear equations, e.g., the Newton-Raphson technique or one of the search techniques which have been widely applied for such problems.

Although the least squares curve-fitting method is most frequently used, it is not always the most desirable. In some situations one wishes to fit the data by a curve which minimizes the greatest distance between the fitted curve and the given data, whereas the least squares method minimizes the sum of squares of the distances. For example, if the data are precise in the sense that they are results of some mathematical calculation (such as the solutions of a differential equation at a particular value of the independent variable) it may be desirable to relate the solution, which may be a performance measure of interest, to the values of certain design parameters in order to reduce the need for solving the differential equations many times.

As an example, in the design of nuclear reactors a problem of importance to the design engineer is the hot spot in sandwich-type fuel elements which contain a

uranium alloy as the center section and another alloy for the external plates. The differential equations used in solving for the maximum temperature are quite involved and require considerable computing time on a modern high-speed computer. Consequently, it is desirable to make use of solutions of these equations for several parameters to infer what the solution is for other parameter values. The solutions to the equations are exact subject to discrepancy in the model. Thus it is not as meaningful in this case but to minimize the sum of squares of deviations between the fitted curve and the given data as it is to minimize the largest absolute deviation between the two. A linear programming technique can be used to solve the problem for linear approximations.

Optimization

The basic problem is:given $y = f(\underline{x})$, $\underline{x} = (x_1, x_2, \ldots, x_n)$ in some region R, to determine the value of $\underline{x}$ that minimizes or maximizes y.

This is a common problem in analysis; the optimum solution is desired, where optimum is defined by means of an objective function such as cost, reliability, or performance as a function of system design parameters. In general the x_i, $i = 1, \ldots,$ n are not only confined to some region, but particular functions of the x_i must satisfy given design constraints. The form of the objective function and that of the constraint function dictate the type of procedure(s) that apply. For example, if the objective and the constraint functions are both linear, a linear programming (LP) approach can be made. If the objective function is quadratic (nonlinear), then a quadratic (nonlinear) programming technique will be used in determining the optimum parameter values. If there are no constraints, such as in the case of the least squares equations for nonlinear models, search techniques or gradient techniques are used in most situations. See Ref. 2-2 for a further discussion of these procedures.

The following table contains a listing of optimization programs categorized by the mathematical problem area such as described above. Additional literature references concerning the particular programs are noted after the program identification number. The prefix to the number when present indicates the machine configuration. Because a large number of LP programs are available no attempt is made to give a complete listing of these. However, for the remaining categories of programs the listing should be reasonably complete with the exception of programs for dynamic programming and the analytical techniques of differential calculus and calculus of variations.

In the case of dynamic programming it is only possible to write programs which solve a particular type or class of problem, such as a reliability optimization problem. If the problem can be solved by methods of differential calculus, then the analytical problem becomes one of solving the resulting system of equations for the location of the stationary points and hence of testing the nature of the function or the matrix

Table 2-1

Listing of Optimization Programs by Mathematical Programming Problems

OBJECTIVE FUNCTION

CONSTRAINT FUNCTION	(1) Linear	(2) Quadratic	(3) Separable(Stagewise)	(4) Non-linear-Not (2) or (3)
Linear	Linear Programming . Deterministic . Integer . Stochastic - - - - - - - - - 7040-C0-12X [Ref.2-8] 7094-K1 3206M3 [Ref.2-8] 7040-H1 3384LSOB [Ref.2-8] 3600-15.2.001 [Ref.2-8] LIP 1 SHARE (SDA3335) [Ref.2-10] IP01,2,3 SHARE(1192,1191 and 1190) [Ref.2-10]	Quadratic Programming 7040-H1 3326QPF4	Dynamic Programming Many programs cited in the literature for specific problems; see Refs. 2-8 through 2-12.	Non-linear Programming 7040-H9 IBM 0007 [Ref.2-8] 7090-H9 IBM 0021 [Ref.2-8] 7090-H2 3430GPGO [Ref.2-8] 7040-H2 3429GP40 [Ref.2-8] 7040-H2 3189SORT [Refs.2-8 and 2-9] 7090-H1 3199NLP [Ref.2-8]
Nonlinear	Non-Linear Programming 7094-K1 3206M3 [Ref.2-8] (See column (4) - Nonlinear Programming Problems -- Calculus of Variations).			Calculus of Variations
No Constraint		Differential Calculus		Search Techniques 7090-H0 3214MINS [Ref.2-8] 0709-C3 3376SEAR [Ref.2-8] MINI [Ref.2-11] BOTM [Ref.2-11] FIBONACCIAN [Ref.2-7] DIRECT SEARCH [Ref.2-7] ROSENBROCK [Ref.2-12] SCOOP [Ref.2-11]

of second derivatives at each of the extremal points to determine whether it is a maximum, a minimum, an inflection, or a saddle point. These calculations can all be performed numerically if desired. No programs are identified in this area. Similar treatment of the method of Lagrangian multipliers is possible for some problems with constraints. However, if a linear, quadratic,or nonlinear programming technique is applicable, the method of Lagrangian multipliers is probably not going to be efficient.

Ref. 2-2 contains summaries of several publications in which one or more of the optimization procedures are applied.

Simulation

The problem statement is: given a process or system which yields an output y for given inputs $\underline{x}$, characterize the output y. One approach to describing an output y is to simulate the process by generating the inputs $x_1, \ldots, x_n$ by an appropriate procedure, such as the use of a random number generator, and then use a system model to obtain the output. This procedure is repeated a sufficient number of times to characterize the output to the degree of precision desired.

A great variety of problems can be solved by simulation. For example, if $y = f(\underline{x})$ is a complex function of random variables $x_1, \ldots, x_n$, then the distribution of the random variable y can be estimated by performing a sufficient number of Monte Carlo runs. Such a procedure is often used in reliability and parameter variation analysis as a means of estimating the probability that the performance measure of interest will fall inside certain limits; such Monte Carlo techniques are discussed later in the report.

Simulation can be applied to random walk problems, such as that of a neutron particle in a nuclear reactor, to the behavior of a sequential test procedure given certain assumptions concerning the underlying distributions, or to diffusion problems. An industrial process can be simulated for the purpose of improving the efficiency. Repair and service time (queueing) problems are examples which may require the use of simulation techniques. Of course many of the above problems, if sufficiently simple, can be treated analytically and the use of a Monte Carlo procedure is wasteful. In many real world applications, however, the complexity is such that the use of approximations or a simulation is required.

Differentiation

The problem is: given $y = f(\underline{x})$, determine $\frac{\partial y}{\partial x_i}$, $\frac{\partial^2 y}{\partial x_i \partial x_j}$, $\frac{\partial^2}{\partial x_i^2}$, etc.

This problem can be treated by the appropriate combination of the techniques given above. However, it is a basic problem of frequent application and uses the techniques of difference calculus. For example, one obvious procedure for obtaining the first derivative of a given function at point x_1 is to evaluate the function at three

equally spaced points x_o , x_1 , x_2 and average the corresponding slopes of the secant lines connecting the points as shown in the following figure. Thus the estimate of the derivative is

$$\frac{dy}{dx} \simeq \frac{1}{2}\left[\frac{y_1-y_o}{h} + \frac{y_2-y_1}{h}\right] = \frac{1}{2h}\left[y_2 - y_o\right]. \qquad (2\text{-}7)$$

This is a central difference formula; clearly many other such formulas can be obtained. Similarly one can obtain a formula for a mixed or pure second partial derivative. For example, Ref. 2-13 contains many such formulas. It is worth noting that numerical differentiation, interpolation from a set of tables, and the numerical quadrature formulas used in the construction of tables or for looking up values in tables have much in common.

The problem of differentiation can occur in many ways in engineering analysis problems. We may wish to perform a sensitivity analysis in which the relative changes in the performance measures are needed corresponding to changes in each of the independent variables; we may be searching for an optimum and require the gradient of the function f(x); or we may wish to expand a function in a Taylor series to obtain a simple approximating function.

Integration, Definite and Indefinite

The problem is: given the function f(x), determine $F(x) = \int_a^x f(u)du$.

Since integration is the inverse of differentiation, the same basic techniques, again starting with the difference equations, are required. For example, the well

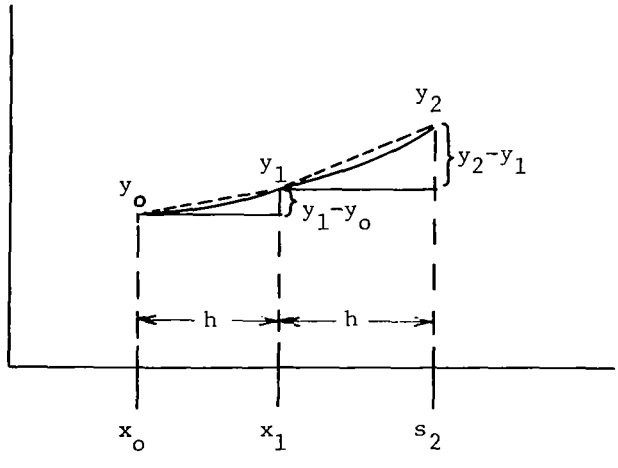


Figure 2-5. Estimation of the Derivative, dy/dx

known trapezoidal rule is used to obtain a definite integral of f(x) over an interval [a,b] as shown in Fig. 2-6.

$$\int_b^a f(x)dx \cong \sum_{i=1}^{n} A_i = \sum_{i=1}^{n} \frac{1}{2}(y_{i-1} + y_i)h$$

$$= \frac{h}{2}\left[y_o + 2y_1 + \ldots + 2y_{n-1} + y_n\right] \qquad (2\text{-}8)$$

More precise formulas for the definite integral can be obtained by using second degree approximations (Simpson's rule) and higher degree polynomical approximations. Ref. 2-4 contains several such formulas.

In the case of an indefinite integral and differential equations, it is typical to use the difference formulas and Taylor series approximations to estimate the integral function step by step over a given interval starting with known values given by boundary conditions.

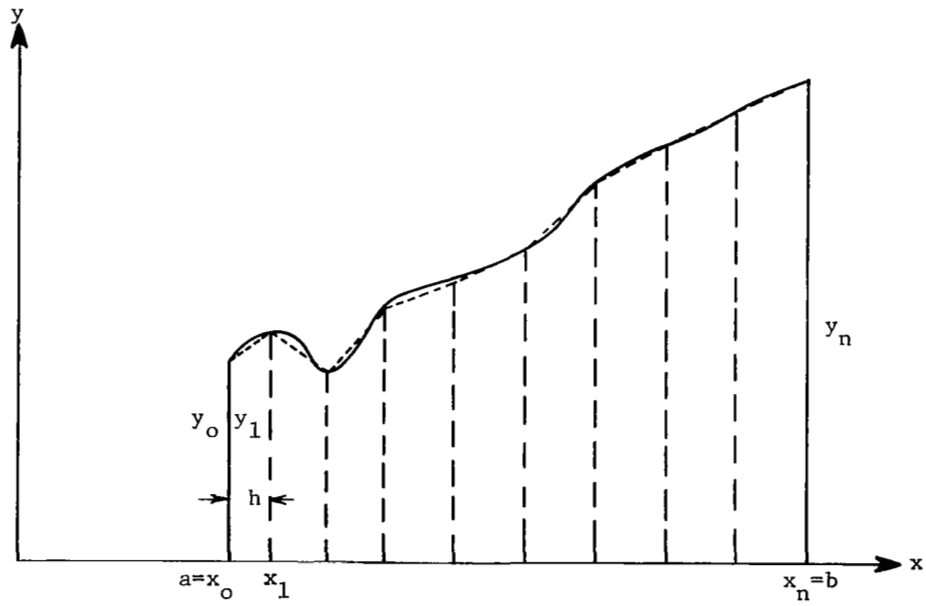


Figure 2-6. Numerical Integration

References

2-1. Computer Design Publishing Corporation: Computer Industry Annual 1967-1968. West Concord, Mass., 1967.

2-2. Katz, D. L., et. al.: Computers in Engineering Design Education, Vol. I - Summary. University of Michigan, College of Engineering, April 1, 1966.

2-3. Wilde, Douglass J.: Optimum Seeking Methods. Prentice-Hall, Englewood Cliffs, New Jersey, 1964.

2-4. Pennington, R. H.: Introductiory Computer Methods and Numerical Analysis. Macmillan, New York, 1965.

2-5. Hamming, R. W.: Numerical Methods for Scientists and Engineers. McGraw-Hill, New York, New York, 1962.

2-6. Hartley, H. O.: The Modified Gauss-Newton Method for the Fitting of Nonlinear Regression Functions by Least Squares. Technometrics, Vol. 3, no. 2, May 1961, pp. 269-280.

2-7. Nelson, A. C., et. al.: Evaluation of Computer Programs for System Performance Effectiveness. Progress Report No. 1 (Lab Project 920-72-1, SF-013-14-03, Task 1604, Contract N00140 66C 0499), Research Triangle Institute, System Statistics Group, October 1966.

2-8. Colville, A. R. Mathematical Programming,"Programs Available From IBM Program Information Department". IBM, Dec., 1966, 20p.

2-9. McCormick, G. P., et al.: Computer Program Implementing the Sequential Unconstrained Minimization Technique for Nonlinear Programming. AD 621 991, April 1965, 87p.

2-10. Balinski, M. L.: Integer Programming: Methods, Uses, Computation. Management Science, Vol. 12, no. 3, Nov. 1965, AD 627 193.

2-11. Shapiro, M. S.; and Goldstein, Max: A Collection of Mathematical Computer Techniques. AEC Computing and Applied Mathematics Center, Courant Institute of Mathematical Sciences, New York University, Feb. 1965, Contract No. AT (30-1)-1480.

2-12. Rosenbrock, H. H.; and Storey, C.: Computational Techniques for Chemical Engineers. Pergamon Press, 1966, 328p.

2-13. Abramowitz, M.; and Stegun, I. A.: Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. Dover Publications, New York, New York, 1965.

3. Reliability and the Computer -- A Perspective

The scope of activity included under the heading of reliability generally can be subdivided into two areas: management and control versus assessment and assurance. The former of these typically includes tasks such as planning, reporting, training, etc. The role of the computer in this area is mainly one of bookkeeping and information storage and retrieval. These uses of computers are not treated in this report.

As a real aid to reliability, the computer's most vital function is in performing complex data processing and analysis operations which prevail mostly in the assessment and assurance activities. These roles are the ones emphasized in this report. The major tasks in which computers can aid reliability with these functions are identified below, then surveyed for a perspective on the role that computers can play in implementing them.

Failure modes and effects analyses (FMEA) are procedures for considering modes of operation of components (such as a short of a resistor or premature operation of a transmitter) and the effects these modes have on system operation. Parameter variation analyses (PVA) treat variations in performance using models (either mathematical or physical) which relate performance to characteristics of the components and operating conditions that cause the performance to vary. Part application analyses consider individually the parts and components of the system for a comparison of operating conditions to rated capabilities. Reliability prediction is concerned with the probability of successful operation of an equipment using models that relate system success probabilities of events associated with components and operating conditions; it can include probabilities related to both life and performance. Testing is concerned with all effects introduced above; it alone can be a means to an end or serve both a supplementary and complementary role to the analyses by supplying information to support the formulation of models, data inputs to them, and checks of their validity.

The first four of these are analysis tasks initiated early in design. A perspective for their coordinated implementation for treating reliability problems in design and development is illustrated in Fig. 3-1, which also includes a general indication of computer utility for performing these tasks. Testing also often employs computer methods and as noted earlier, this task serves as support to the analyses. The proposed design and mission define the problem to be analyzed. The analyses provide the output information for design improvement and assurance. Improvement results through a feedback process whereby the design or mission is modified as required. Such modifications require tradeoffs between reliability and other requirements of the system (cost, maintainability, etc.) before being made.

In brief, the overall objectives of the reliability analyses are:

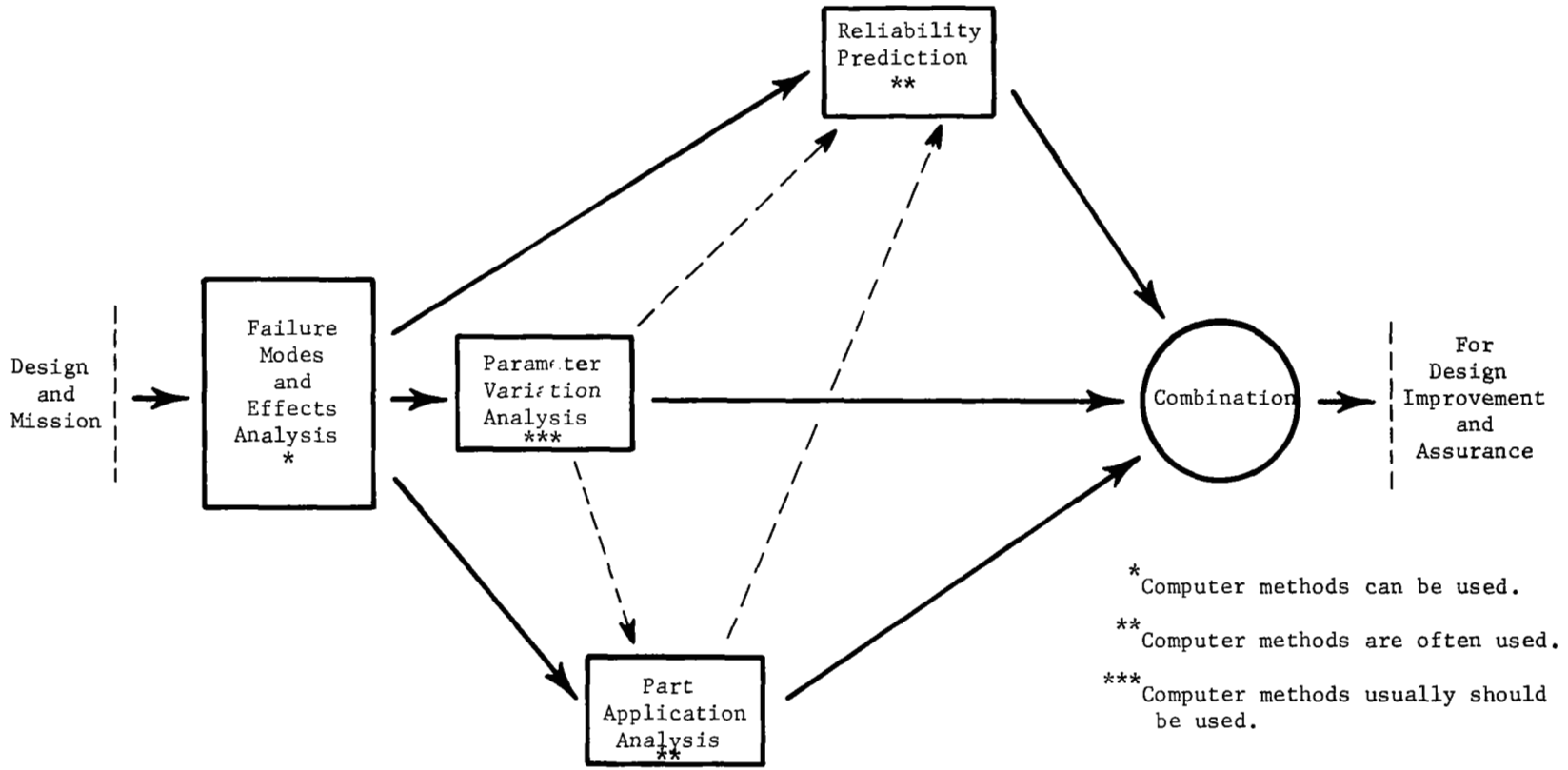


Figure 3-1. Reliability Analyses in Design and Development

(1) identifying and removing possible causes of failure,

(2) balancing safety (design) margins or apportioning tolerances, and

(3) obtaining numerical assessments of reliability.

None of the defined reliability tasks is capable of achieving these objectives by itself. As illustrated, the tasks are strongly interrelated; it is through their coordinated application and the combined use of their results that maximum benefit is derived for reliability. The computer can aid in performing each of the individual tasks; for example, it usually should be used for PVA and often should be for reliability prediction. Each of the tasks and the relevance of computer methods to implementing each are discussed below.

Failure modes and effects analysis serves the purpose of revealing what can happen to the system. By considering the likelihood and the criticality of the possible modes of system behavior, it allows direction of effort in the other reliability tasks. It defines specific modes of behavior for performance variation studies; it identifies critical areas to be emphasized in part application analyses; it designates failed states to be included in reliability predictions. Because of its value in directing other effort, a failure modes and effects analysis should be initiated early in the design program. A computer is seldom used in identifying failure modes; it is used in investigating failure effects as discussed in Sec. 6 of this report.

Parameter variation analysis is concerned with the assurance that performance is acceptable. Whereas reliability prediction, failure modes and effects analysis, and part application analysis are usually formal tasks in system contractor activities parameter variation analysis has been neglected due to limited understanding of the available techniques for treating performance variability.

As described in Sec. 4 a number of analytical techniques have been assembled and tested, and a flexible PVA program has been written. In this program, mathematical or physical models are used to relate performance attributes to component and interface characteristics. Probabilistic techniques such as propagation of moments and Monte Carlo simulation are used to estimate probabilities or distributions of performance. Various end-limit techniques provide worst-case performance values and parameter sensitivities. Sources of variation are identified and relative contributions of component variation can be determined. Parameter variation analyses yield directly useful design information and, as illustrated in Fig. 3-1, provide inputs to the other tasks. These include, for example, operating conditions for components used in application analyses and performance estimates to be included in reliability predictions.

Part application analysis determines whether components are properly applied. For example, thermal and electrical loads on parts are used for appropriate adjustment

of failure rate estimates, and parts with loads exceeding design specifications are appropriately respecified or the design changed to reduce these loads. Computers are readily used to make a part application analysis; such an analysis is frequently conducted as a part of a larger analysis. For example, it is easy when performing a circuit analysis to check actual voltage, current, and power against rated values for each component in the circuit, and provisions for doing this are incorporated in some circuit analysis programs. Further discussion is given in Sec. 5.

Reliability predictions are based on logic relationships expressing success or failure event probabilities of system components. Currently, most prediction calculations are based on two-state (success vs. failure) models using part failure rates and exponential life distributions. Because of the many simplifying assumptions, little significance can be attached to the magnitudes of the numbers obtained. Some advanced techniques consider more than two states as discerned by the failure modes and effects analysis, and more appropriate life distributions are also available. Although the framework has been developed for including performance degradation failures in prediction, the value of reliability prediction at present lies more in the design weaknesses detected in performing the analysis and to compare alternative designs than in the actual numbers resulting. The application of these techniques by computers is treated in Sec. 7. Computers play a valuable role by enabling more realistic prediction models to be employed and by performing the computations which produce the reliability estimates resulting from these models.

Each method above separately provides useful design information, but to assure appropriate emphasis on both performance and life, the results from the various methods must be considered jointly. Because of the different forms of the results the combination process is primarily subjective, so the computer can provide little help here. As an example on the combination of the tasks, suppose that parameter variation analyses have yielded worst-case results for two designs being compared and that Design A has smaller variations than Design B. Reliability predictions with conventional two-state analyses may, in turn, indicate that Design B has a higher probability of success. Indications are thus that Design B represents an improvement in life over Design A, however at a sacrifice of performance. If there is adequate confidence in the results of each, a trade-off may be necessary, for example, resulting in Design C that uses some of the better features of Designs A and B. On the other hand, lack of confidence in the results may dictate the need for more sophistication in the analyses. For example, an extension of prediction to more realistically include additional modes of part failures and their effects may show that Design A is the better from the standpoint of life.

No one of the reliability tasks provides a "cure-all" for reliability,but through their coordinated and combined use, the maximum assurance for reliability is achieved. Also,the responsibility for reliability cannot be delegated to reliability specialists alone. Reliability is a responsibility of all personnel, but the major responsibility rests with the designer. Good engineering is, and will remain, the major key to reliability. The methods are provided as a supplement to, but not a substitute for, good engineering practice.

Just as performing these tasks is no substitute for good engineering, neither is the indiscriminate use of the computer to perform such tasks good reliability engineering. Computer methods should be selectively used in design for reliability, and used only when they can provide genuinely useful results within the economic, time and other relevant constraints on the design under consideration. Within the bounds of these constraints, the computer aids to design for reliability which are discussed in the remainder of this report comprise a powerful set of tools for insuring that a reliable product is produced.

4. Parameter Variation Analysis

There are two ways in which a piece of equipment or a system can fail to perform its intended function. One is catastrophic failure, which is likely to be abrupt and to have a dramatic effect on equipment or system operation. In an electronic circuit, a typical catastrophic failure is the opening or the shorting of a diode or a transistor. The other type of failure is drift failure, where due to the variations of equipment parameters with time, the performance of the equipment at some time becomes no longer satisfactory. The prediction of drift-type failures requires a study of combinations of component parameter values and the resulting effects of the drifting of these values on equipment performance. Studies of parameter drifts and their effects on system performance comprise parameter variation analysis (PVA). The availability of the high speed digital computer has made possible a dramatic increase in the ability to perform extensive PVA studies and as a result improve the reliability of the equipment by minimizing via design modifications the likelihood of a drift-type failure.

4.1 PVA Modeling

A PVA model must be adequately accurate to simulate the equipment behavior over the entire range of environments expected for the equipment*. To enable the PVA analysis to be accomplished, the model must express the relationships between the performance characteristics of interest and all the parameters to be included for study. In many cases, the equipment passes through several distinct operating regions, and it is necessary that the model adequately represent each region. A change from the ON to the OFF state of a transistor, for example, requires a new equivalent circuit for the transistor, and each such equivalent circuit must adequately simulate the actual circuit operation to provide engineering confidence in the performance it pre-predicts.

At the core of any parameter variation analysis is a mathematical model; in explicit form,

$$Y_j(t) = g_j[\underline{X}(t)] \qquad (4\text{-}1)$$

or implicit form

$$g_j[\underline{X}(t), Y_j(t)] = 0,$$

where

$Y_j(t)$ is the jth performance attribute or measure,

---

*Some very practical viewpoints on modeling are presented in Vol. I - Parameter Variation Analyses of this series.

$X(t)$ is a vector comprised of the environment inputs, such as environmental stresses and loads, plus the component characteristics,

t is the time variable, and

$g_j, j=1,\ldots,N$ is the set of models corresponding to the number of responses or the order of the differential equations which describe the transient behavior of the system.

For example, the model may be of the form of a system of differential equations,

$$\frac{\partial^2 Y}{\partial t^2} + c_1 \frac{\partial Y_1}{\partial t} + c_2 Y_2 = c_3$$

$$\frac{\partial Y_2}{\partial t} + c_4 Y_2 + c_5 Y_1 = c_6 ,$$

where the c_i depend on the input vector $\underline{X}$ through a set of explicit expressions.

The time behavior for the model may appear in one of several ways. For example, it may be a gradual deterioration of a component and hence result in a corresponding change in the values of one or more of the component characteristics. In order to analyze an element or system for this type of degradation, the wearout characteristics of the system must be known or estimates must be available.

A second way in which time may appear is through the mission profile. For example, if it is known that the temperature profile is critical and how the part characteristics vary with temperature, then an analysis can be performed by describing the temperature-part characteristic behavior by deterministic and/or random processes and performing the analysis at several times in the mission life.

Time may enter the analysis directly through the transient behavior. In this case a program for solving differential equations may be required for relating the transient characteristics to the pertinent element parameters, inputs, etc. In whatever manner time enters the analysis, it is assumed that it may be included by a procedure such as one of the following:

(1) A deterministic function of time such as a linear or exponential decay function.

(2) An autoregressive scheme such as

$$X_{jt} = A_1 X_{j,t-1} + A_2 (X_{j,t-1} - X_{j,t-2}).$$

(3) A stochastic process such as a normal stationary process superimposed on a deterministic drift.

(4) A system of differential equations.

4.2 Analysis Techniques

Several analysis techniques are used for PVA on a computer. One of the most widely used is worst-case analysis. The worst-case method is a nonstatistical approach which is intended to determine whether it is possible, within the specified tolerance limits on parameters, for the system performance to fall outside the specified performance limits. The answer is obtained by using performance models, and setting the parameter values at combinations of upper and lower tolerance limits to obtain the worst-case performance. A related PVA technique is sensitivity analysis. Worst-case and sensitivity analyses typically use the same mathematical techniques, as is discussed later. The purpose of a sensitivity analysis is to determine how sensitive a system performance variable is to variations in input variables.

Another common technique for performing PVA is the moments method. This technique combines statistics and system analysis to determine the probability that performance will remain within the specified limits; the technique is often called the propagation-of-variance method as second moments of distributions are usually the highest moments used. The method applies the propagation-of-variance formula to the first two moments of the component part probability density functions to obtain the equivalent moments of the performance distribution.

The convolution method for PVA is another approach to obtaining statistical distributions of output variables. Although potentially a quite general method, the technique reported here and as implemented by computer programs is a simplified version of the general convolution method.

In the Monte Carlo method component values are selected randomly; the performance of each randomly generated configuration of the equipment under study is calculated and compared with performance specification limits. This technique has the advantage that any component parameter distribution can be handled; it has the disadvantages that it requires a lot of computer time and offers little help in identifying and correcting failures.

The implementation on a computer of each of the above techniques is now treated in detail. Some of the computer programs which are available for implementing the techniques are discussed.

4.2.1 Worst-Case Analysis

The theory on which worst-case analysis is based derives from expressing the model performance parameters Y_j as functions of the input vector $\underline{X} = (X_1, X_2, \ldots, X_m)$ and expanding these functions in Taylor series. The input vector $\underline{X}$ consists of all pertinent part characteristics, inputs, loads, and environment factors. Let the model for an arbitary performance parameter Y be

$$Y = g(\underline{X}) = g(X_1, X_2, \ldots, X_m). \tag{4-2}$$

A Taylor series expansion about the nominal value of Y for its change from nominal value is

$$\Delta Y \cong \sum_{i=1}^{m} \frac{\partial Y}{\partial X_i}\bigg|_{\underline{X}_N} \Delta X_i + \frac{1}{2} \sum_{i=1}^{m} \frac{\partial^2 Y}{\partial X_i^2}\bigg|_{\underline{X}_N} (\Delta X_i)^2 + \ldots , \tag{4-3}$$

where

ΔY = change in value of Y from its nominal value,

$\Delta X_i = X_i - X_{i_N}$, the worst-case deviation of the i-th independent variable X_i from its nominal value X_{i_N} , and

$\underline{X}_N = (X_{1_N}, X_{2_N}, \ldots, X_{m_N})$, the nominal values of the X's.

Eq. (4-3) is a simplified expansion which includes no cross-product terms; a completely general Taylor series expansion is given in Appendix B, Vol. I - Parameter Variation Analysis of this series. In practice the cross-product terms are seldom used even in computer programs, so Eq.(4-3) is the expansion most likely to be found. Frequently only the linear terms are used; the expansion then has the familiar form

$$\Delta Y \cong \frac{\partial Y}{\partial X_1} \Delta X_1 + \frac{\partial Y}{\partial X_2} \Delta X_2 + \ldots + \frac{\partial Y}{\partial X_m} \Delta X_m . \tag{4-4}$$

To perform a worst-case analysis, the partial derivatives of Y with respect to each independent variable X_i must be computed. Several techniques are used to compute derivatives on a computer. The "Eight Point Central Derivative Formula" is a popular method [Refs. 4-1 and 4-2]. This formula for the first partial derivative is

$$Y_i' \equiv \frac{\partial Y}{\partial X_i}\bigg|_{\underline{X}_N} \cong \frac{1}{hX_{i_N}} \left[\frac{4}{5} (Y_{+1h} - Y_{-1h}) - (Y_{+2h} - Y_{-2h}) + \frac{4}{105} (Y_{+3h} - Y_{-3h}) - \frac{1}{280} (Y_{+4h} - Y_{-4h})\right]. \tag{4-5}$$

This formula is evaluated by stepping the input parameter X_i four equal increments h each way from its nominal value X_{i_N} , and calculating the value of Y for each step while holding all other independent variables X_j, $j \neq i$, at their nominal values.

In this formula, h is expressed as the fractional change in X_i; if one percent of X_{i_N} is the step size, h is 0.01. The values of Y are then substituted into Eq. (4-5) and the partial derivative $\partial Y/\partial X_i$ obtained. This method is used in the worst-case analysis method called MANDEX,* which is perhaps the most widely used worst-case computer method [Ref. 4-2].

A second formula for computing partial derivatives via computer is the five-point central difference formula [Ref. 4-3]. The first derivative formula is

$$Y_i' \equiv \frac{\partial Y}{\partial X_i}\bigg|_{\underline{X}_N} \approx \frac{1}{12h}\left(Y_{-2h} - 8Y_{-1h} + 8_{+1h} - Y_{+2h}\right). \tag{4-6}$$

This is evaluated analogously to the eight-point one, but its accuracy is somewhat less. However, its accuracy usually is adequate when only the first derivatives are used in the Taylor series expansion. A five-point formula for the second partial derivative with respect to one independent variable is

$$Y_i'' \equiv \frac{\partial^2 Y}{\partial X_i^2}\bigg|_{\underline{X}_N} \approx \frac{1}{12h^2}\left(-Y_{-2h} + 16Y_{-1h} - 30Y_o + 16Y_{+1h} - Y_{+2h}\right); \tag{4-7}$$

these five-point equations are used in one of the PVA program discussed later. Eqs. (4-6) and (4-7) are derived in Abramowitz and Stegun [Ref. 4-4]; note that in these equations h is just a number, not a fraction of X_{i_N}.

Having evaluated the partial derivatives, the worst-case limits are next computed. The signs of the first partial derivatives are examined to enable a procedure for computing worst-case limits which reduces computing time. A worst-case maximum by definition occurs when the performance parameter Y takes on its greatest value, i.e., when ΔY is maximum and positive. Consequently, all input variables with positive first partials are set at their upper limits and all with negative first partials at their lower limits. This procedure gives the worst-case maximum in the linear Taylor series expansion, Eq. (4-4), since each term is a product of either two positive or two negative quantities. For the worst-case minimum, lower limits are used for the X_i with positive partials and upper limits for those with negative partials, producing all negative terms and hence the worst-case minimum in the linear series summation for ΔY.

It is possible that the partial derivative of an output variable Y with respect to an input variable X_i is not linear; then the above procedure does not necessarily

---

*MANDEX is an acronym for modified and expanded worst-case analysis.

produce true worst-case limits. Linearity checks or more complex series expansions can be incorporated to prevent such inaccuracies from going unnoticed. These safeguards are discussed in Sec. 4.2.5 as to how they are implemented in specific PVA programs.

Worst-case analysis is applied most widely to electronic circuits, but it is equally applicable to any system for which a performance model can be derived and input parameter variations are known or can be reasonably estimated. The proper use of worst-case analysis is as a first step in the PVA study of a system. If the system passes this parameter variation analysis, it is almost certain to pass any other. Hence it is possible to accept a design if it passes worst-case analysis. Conversely, it usually is in error to reject the design only because it fails a portion of a worst-case analysis, since the probability of obtaining a true worst-case condition in practice is very small. A failure to pass a worst-case analysis usually indicates that other analyses should be performed.

4.2.2 Sensitivity Analysis

An important PVA technique related to worst-case analysis is analysis of the sensitivity of system performance to variations in input parameters. Although several different definitions of sensitivity are found in the literature [Refs. 4-3 and 4-5], in essence the sensitivity of a system is simply a measure of the effect of parameter variations on the system performance. In equation form sensitivity can be expressed by

$$S_{X_i}^{Y_j} = \Delta Y_j / \Delta X_i \ , \tag{4-8}$$

where

$S_{X_i}^{Y_j}$ is the sensitivity of the performance measure Y_j to the variation in the system model parameter X_i,

ΔY_j is the change in Y_j, and

ΔX_i is the variation in X_i.

An alternative form is the normalized sensitivity

$$S_{X_i}^{Y_j} = \frac{\Delta Y_j / Y_j}{\Delta X_i / X_i} \ ; \tag{4-9}$$

it is more frequently used.

Each of the terms on the right side of Eq. (4-9) is either available or easily obtained from the performance model. All that is required to obtain sensitivity is to calculate ΔY_j (the change in Y_j produced by the change in X_i only) and then perform

the three arithmetic operations indicated in Eq. (4-9) for each performance variable Y_j and input variable X_i. The combination of worst-case and sensitivity information on a design is complementary, particularly when design modifications are required. Suppose a design fails to pass a worst-case analysis for a performance measure Y_j with respect to a variable X_i. If also the sensitivity $S_{X_i}^{Y_j}$ is high, e. g., a 1% change in X_i produces a 5% change in Y_j, a redesign around the variable X_i may be needed. If worst-case analysis with respect to X_i fails for several output variables Y_j and the corresponding sensitivities are high, such a redesign probably is required.

The accuracy of a sensitivity calculated with Eq. (4-9) is obviously limited by the accuracy of the assumptions and approximations used in the calculation. For example, maximum sensitivity may occur somewhere between, rather than at, the upper and lower input parameter limits. The remarks made for worst-case analysis on linearity and higher order series expansions also apply here.

4.2.3 Moments Analysis

The moments method of PVA analysis has this name because it makes use of the moments of the statistical distributions of input parameters to obtain the moments of the distributions of the system performance measures. As usually implemented on a computer, it makes use of the first moment (the mean) and the second moment about the mean (the variance) of the distributions of the input parameters to obtain the mean and the variance of the distributions of the system performance measures. When a distribution is normal these two moments describe it completely. Although distributions which are found in practice are seldom precisely normal, the accuracy is often adequate for PVA purposes. This simplified form of the moments method, called the propagation-of-variance method, is what is described below.

The mean values for the model output parameters are obtained by programming the computer to insert mean values for all the variables in the system model input vector and then solve the performance equations. The computer then calculates the second moment about the mean, i.e., the variance,of each output variable by evaluating the propagation-of-variance formula given below. An additional feature incorporated in some programs is that each of the terms in the propagation-of-variance formula is divided by the total variance to give an indication of the fraction of the variance contributed by each input parameter.

The propagation-of-variance formula is the heart of the computer-implemented moments method of analysis. This formula is the mathematical statement that the performance variability is the net result of the variability of all the input parameters in the system, and that the contribution of each input parameter depends upon its individual variability and on the relative importance of that parameter in

determining the performance characteristic of interest. The propagation of variance formula is

$$\sigma_i^2 = \sum_{j=1}^{N} \left(\frac{\partial Y_i}{\partial X_j}\right)_{\bar{X}_j}^2 \sigma_{X_j}^2 + 2 \sum_{r=1}^{N-1} \sum_{s=r+1}^{N} \rho_{rs} \sigma_{X_r} \sigma_{X_s} \left(\frac{\partial Y_i}{\partial X_r}\right)_{\bar{X}_r} \left(\frac{\partial Y_i}{\partial X_s}\right)_{\bar{X}_s} \qquad (4\text{-}10)$$

where

σ_i^2 is the variance of the performance parameter Y_i,

$\sigma_{X_j}^2$ is a variance of the input parameter X_j,

N is the number of contributing input parameters, and

$\bar{X}_j$ is the mean value of X_j.

The term ρ_{rs} is a correlation coefficient that relates the parameter contributions X_r and X_s, and the subscripts ($\bar{X}_j$, $\bar{X}_r$, and $\bar{X}_s$) indicate the points at which the partial derivatives for these input parameters are obtained.

The first term in Eq. (4-10) includes the variance of each input parameter and the partial derivative of the performance measure with respect to that parameter. Since the factors in this term are squared they are all positive. The second term in the equation can be either positive or negative; it includes each pair of correlated parameters. This term simulates the true situation in which correlation between two input parameters can either increase or decrease the total performance variability. From this equation the variance of any performance measure Y_j can be obtained from knowledge of the mean, variance, and correlation coefficients of each input parameter.

In the propagation-of-variance method all output variables are assumed to be linear functions of the input variables, and all input parameter distributions are assumed to be normal. Hence, non-normal input parameter distributions are approximated by normal ones in the propagation-of-variance formula. As seen from Eq. (4-10), the method requires the calculation of partial derivatives. This can be done in precisely the same way that the partial derivatives are calculated for worst-case analysis.

Possible sources for values of moments of the input parameter distributions are manufacturer's data, testing a large number of components, or assumptions based on experience. For example, recording and plotting the resistance values of a large number of resistors of a given nominal value will produce a plot, known as a histogram, as shown in Fig. 4.1. In the figure the widths of the small rectangles, called cells, represent equal increments of resistance values that fall within the individual

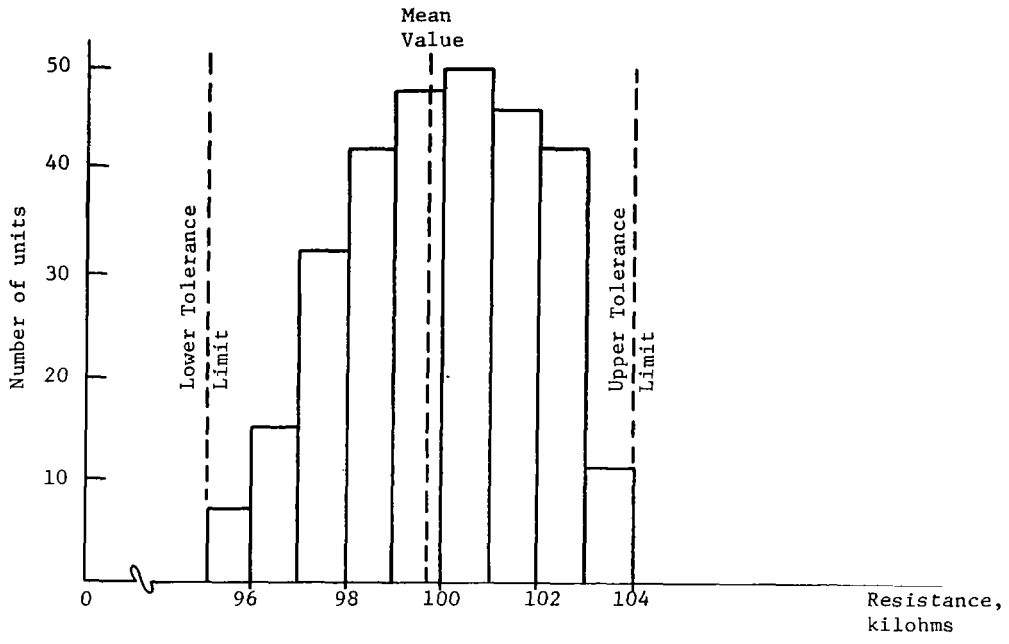


Figure 4-1. Histogram of Resistance Values for a Resistor

resistance increments. The sum of the heights of all the cells equals the total number of resistors tested. A mean value for this input parameter, namely resistance, can be calculated by adding together the resistances of the individual units and dividing the sum by the total number of units. The variance, σ^2, is calculated by taking the number of resistors in each cell and multiplying each by the square of the difference between the midcell value and mean value; these products are then added and divided by the total number of resistors to give σ^2. The square root of the variance, σ, called the standard deviation, is frequently used to discuss the dispersion of normal frequency distributions.

4.2.4 The Convolution Method

The convolution method is another attempt to account for the statistical distribution characteristics. The simplified form discussed here also relies on the partial derivatives as computed above. This approach has generally found only limited practical application; a computer implementation and comparison with other techniques is described in Ref. 4-6.

The convolution method described in Ref. 4-6 is a specialization of the more general analytical approach described in Ref. 4-7. The basis for the convolution method is the assumption that the total variation in an output performance parameter is the sum of the deviations caused by each input parameter independently. This is analogous to the assumption that no mixed product terms of the Taylor series are

required. When limited to linear terms only, the partial derivatives represent the functional relationship between each individual parameter contribution and the parameter variations. For a particular interval of the total range of variation of the output performance parameter, the corresponding interval of each of the model parameters can be determined by obtaining the inverse of the partial derivative. The probability that the output parameter lies in a particular interval is the sum of the relative probabilities for the individual input parameters in their respective and corresponding intervals. Repeating this process over the appropriate intervals to cover the total range of variation yields a histogram representation for each output parameter. Since the convolution method does not assume normal distributions, it can be used to advantage when input parameters are known to have distributions differing significantly from normal.

4.2.5 Monte Carlo Analysis

The theory of the Monte Carlo approach to PVA is based on a statistical theorem called the Glivenko-Cantelli theorem [Ref. 4-2] which is:

> Given a function of n random variables, $Y = f(X_1, X_2, \ldots, X_n)$ with each variable X_i described by a distribution, then select a value for each X_i, $i=1, 2, \ldots, n$, from their respective distributions and compute a value of Y. Repeat this procedure for m times. As m tends to infinity, the distribution of Y obtained approaches the actual distribution of Y.

In contrast to worst-case analysis which obtains only end-limit values and to propagation-of-variance analysis which assumes normal distributions only, a Monte Carlo analysis determines the actual statistical distributions of the output variables. The Monte Carlo method permits computer simulation of a brute-force empirical approach. The empirical approach would require the actual construction from representative components of many copies of the system under study. As many copies would be made and operated as required to obtain good statistical estimates of the system output variables and the variations in these variables. This empirical approach is usually highly impractical, and it is seldom if ever applied.

By using a digital computer to simulate the above empirical technique, many of the objectionable features are removed. Given the mathematical model of a system under study and a description of the component part populations, it is possible by doing enough simulations to obtain to any reasonable degree of accuracy the distributions of the performance measures. The Monte Carlo method requires the complete statistical distributions of the input variables at some particular time t. The computer randomly selects a value for each input parameter from its distribution and uses this value in computing the solution. The values from a multi-parameter part cannot be given simply as distributions. Instead they must be listed in a way

such that all the parameter readings from the same part are grouped together with any necessary correlations. Then the Monte Carlo method makes a single random selection from this listing which determines all the correlated parameter values for the multi-parameter part.

In the Glivenko-Cantelli theorem, each of the random variables X_i, which are in this case the system model input variables, can have either a continuous probability density function or a discrete probability density function. Because only discrete quantities can be used in computer, only discrete probability density functions are of interest to the Monte Carlo method of analysis. A discrete probability density function is simply a normalized histogram. Shown in Fig. 4.2 is the normalized version of the histogram of Fig. 4.1.

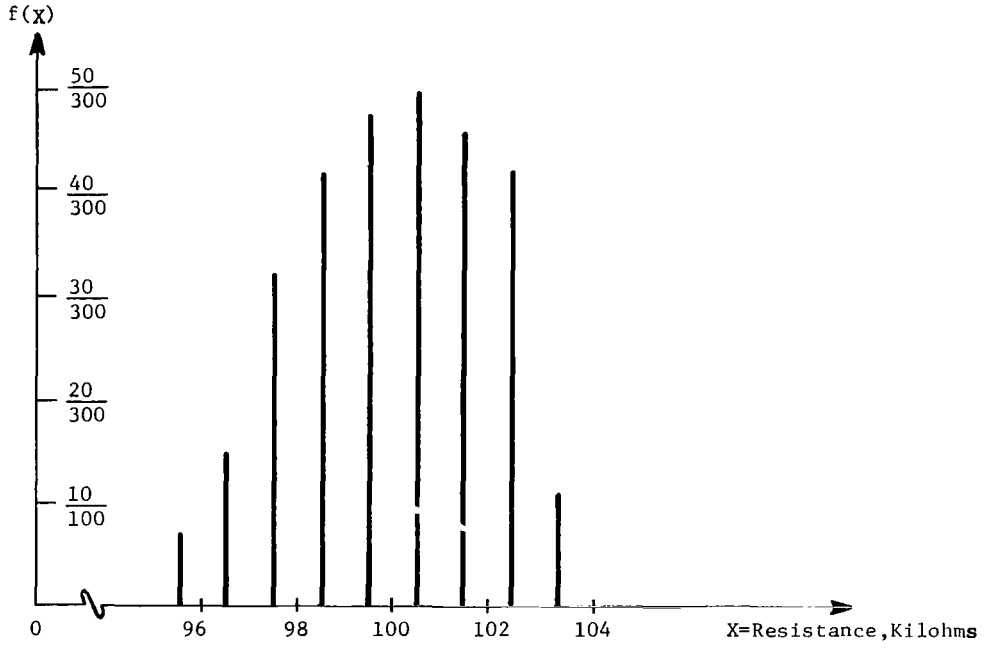


Figure 4-2. Normalized Form of Fig. 4-1; Probability Density Function for Discrete Random Variable X

Given a discrete distribution of associated(f(X), X) values as shown in Fig. 4.2, the discrete random variable X possesses the properties:

Given the values $X = X_a$ and $X = X_b$, $p[X_a \leq X \leq X_b]$ is the probability that $X \geq X_a$ and $X \leq X_b$, where $X_a \leq X_b$.

Also,

$$F(X_n) = \sum_{i=1}^{N} f(X_i) = p[X \leq X_n],$$

where $F(X_n)$ is a point on the discrete cumulative distribution. The summation applies to those values of the random variable X which are less than or equal to the X_n specified in the summation. The cumulative distribution for the random variable X of Fig. 4.2 is shown in Fig. 4.3.

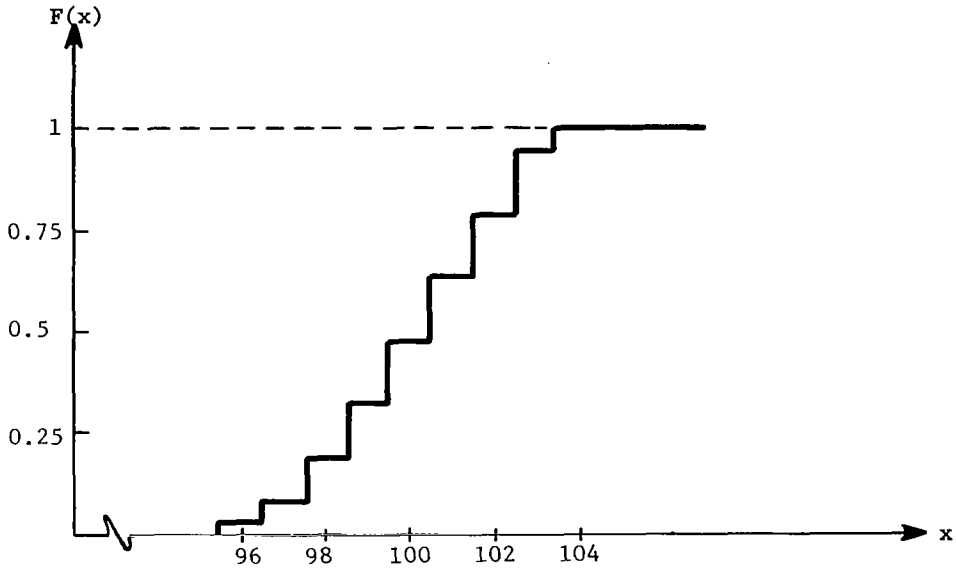


Figure 4-3. Cumulative Distribution for Discrete Random Variable X of Fig. 4-2

An important point to understanding the Monte Carlo method is the observation that an area under the probability density curve amounts to a point on the cumulative distribution curve. Thus, given a density function f(X) for which the total area between X=0 and $X=X_n$ is 0.2, then the value F(X) at $X=X_n$ is 0.2. In a Monte Carlo computer program the computer converts all the probability density functions to cumulative distributions. Then the computer generates random numbers and associates each of these numbers with a particular point on each cumulative distribution. The random numbers in this context are numbers chosen at random in the range between 0 and 1.

In order to obtain reasonable accuracy with the Monte Carlo technique a large number of randomly-generated replicas of the system are made;for the solution to each replica the

output parameters of interest are obtained from the system model equations. The total number of system solutions (also called system simulations) required is obtained via a tradeoff between accuracy and the cost of computer time. This number can vary anywhere from 50 to 5,000 or more depending on the particular application. The number of solutions typically used for one program is 500 [Ref. 4-2]. When practical, a profession statistician should be consulted on how to arrive at an appropriate number of simulations for a given system and purpose.

Once all the Monte Carlo solutions have been generated, the probability density functions for each of the output performance measures can be obtained. Since the complete distribution for each output variable is available, coefficients which describe the various statistical properties of the distributions can be computed as required.

It should be recognized that due to the large number of system simulations required, the Monte Carlo technique is best suited to variation analysis of systems which cannot be handled by less brute-force techniques. Its cost and time limitations must be considered before choosing the technique for a particular system. In a circuit analysis program, for example, dc solutions can be obtained at reasonable cost via the Monte Carlo technique; Monte Carlo ac solutions are usually less practical. Finally, the relatively large amount of computer time required for a single transient solution of a circuit means that it is unreasonable to attempt to obtain sufficient transient solutions to make the Monte Carlo technique a practical approach to obtaining distributions of circuit performance measures related to transient responses.

An interesting variation on the Monte Carlo technique has been reported [Ref. 4-2]. It combines portions of worst-case and Monte Carlo analysis. Often the data giving the actual distributions of input parameters are not available. What has been done in the cited reference for such cases is to substitute a rectangular distribution whose upper and lower limits are the upper and lower worst-case limits. A Monte Carlo analysis is then performed, which provides a better estimate of circuit performance than would be obtained by using the conventional worst-case analysis. Since the actual input parameter distributions are not rectangular, the probability of selecting values close to the worst-case values is greater than for the actual distributions. Consequently, the resulting distributions are less optimistic than would be obtained from the actual input distributions but are not as pessimistic as worst-case solutions.

4.3 PVA Computer Programs

Many computer programs exist for implementing individually the PVA techniques discussed in Sec. 4.2. Some of these programs are listed in Table 4-1. However, relatively few are known to exist which are available outside the organizations

Table 4-1

Programs in the PVA Area

Program Code	Program Description	Organizations (Originator or User/Sponsor)	References
PV-RTI	Performance Variation analyses; general program for worst-case, moments, simulation, etc.	RTI/NASA	4-3
MCS-IBM	Monte Carlo Simulation for performance variation analysis with programmed functional model	IBM/AF-RADC	4-8
MCS-GDC	Monte Carlo Simulation for performance variation analysis with programmed functional model	GD-Convair/?	4-9
PV-LS	Performance Variation analysis program for systems	Lear Siegler/NASA	4-10
PV-SE	Performance Variation analysis program using Monte Carlo simulation with programmed mathematical model	Sylvania Electronics/ AF-RADC	4-11
MANDEX-NAA	Modified AND EXpanded worst-case method for analysis of circuit performance variations with circuit equations	NAA/?	4-2
MM-NAA	Moment Method for circuit performance variation analysis with circuit equations; computer mean and variance; correlation included	NAA/?	4-2
MCS-NAA	Monte Carlo Simulation for circuit performance variation analysis with circuit equations; correlation included	NAA/?	4-2
VINIL-NAA	V_{IN} I_L method for circuit performance variation analysis with circuit equations	NAA/?	4-2
PVM-NAA	Parameter Variation Method for circuit performance variation analysis with circuit equations; one-at-a-time and two-at-a-time analyses	NAA/?	4-2

where they originated and which combine several PVA techniques into a single program. A FORTRAN listing of a general PVA program which implements nearly all of the PVA techniques discussed in Sec. 4.2 is given in Appendix A; it is described in some detail below.

Two widely used circuit analysis programs which have some PVA capabilities are ECAP and NASAP. The Electronic Circuit Analysis Program (ECAP) is available to users of IBM computers. The Network Analysis for System Application Program (NASAP) is a NASA program. Although working at a number of computer installations, NASAP is still in development. These two programs and their PVA capabilities are discussed later in this section.

4.3.1 A General PVA Program

A flow diagram of a general PVA program is shown in Fig. 4-4. As can be seen from the figure, the program is keyed to the subroutine which evaluates the performance model. To make the program applicable to any kind of system, no built-in performance model subroutine is included; this subroutine must be supplied by the user of the program [Ref. 4-3].

The input to the program is a mathematical description of the system model (and the time behavior of the model, if required), the number of random variables and the number of fixed variables involved, the means or nominal values of the input variables, the standard deviations or step sizes in the input variables, the input variable distributions, if available, and the correlations of the input variables. An additional input that is required for some analyses is a selection of values of the element parameters at which the performance model is to be evaluated. If these values are selected methodically according to some statistical design, this allows for efficient generation of the outputs to use in a multiple regression analysis.

Monte Carlo Simulation

A Monte Carlo simulation is used to estimate the performance distribution in terms of the input distributions, characteristics, etc. If the input variables are normally distributed the means, standard deviations, and the correlation matrix are required. If the input variables are not normally distributed the appropriate distribution characteristics must be specified. The program has provisions for handling any one of the following distributions:

(1) Uniform,
(2) Normal,
(3) Log-Normal,
(4) Exponential,
(5) Weibull,

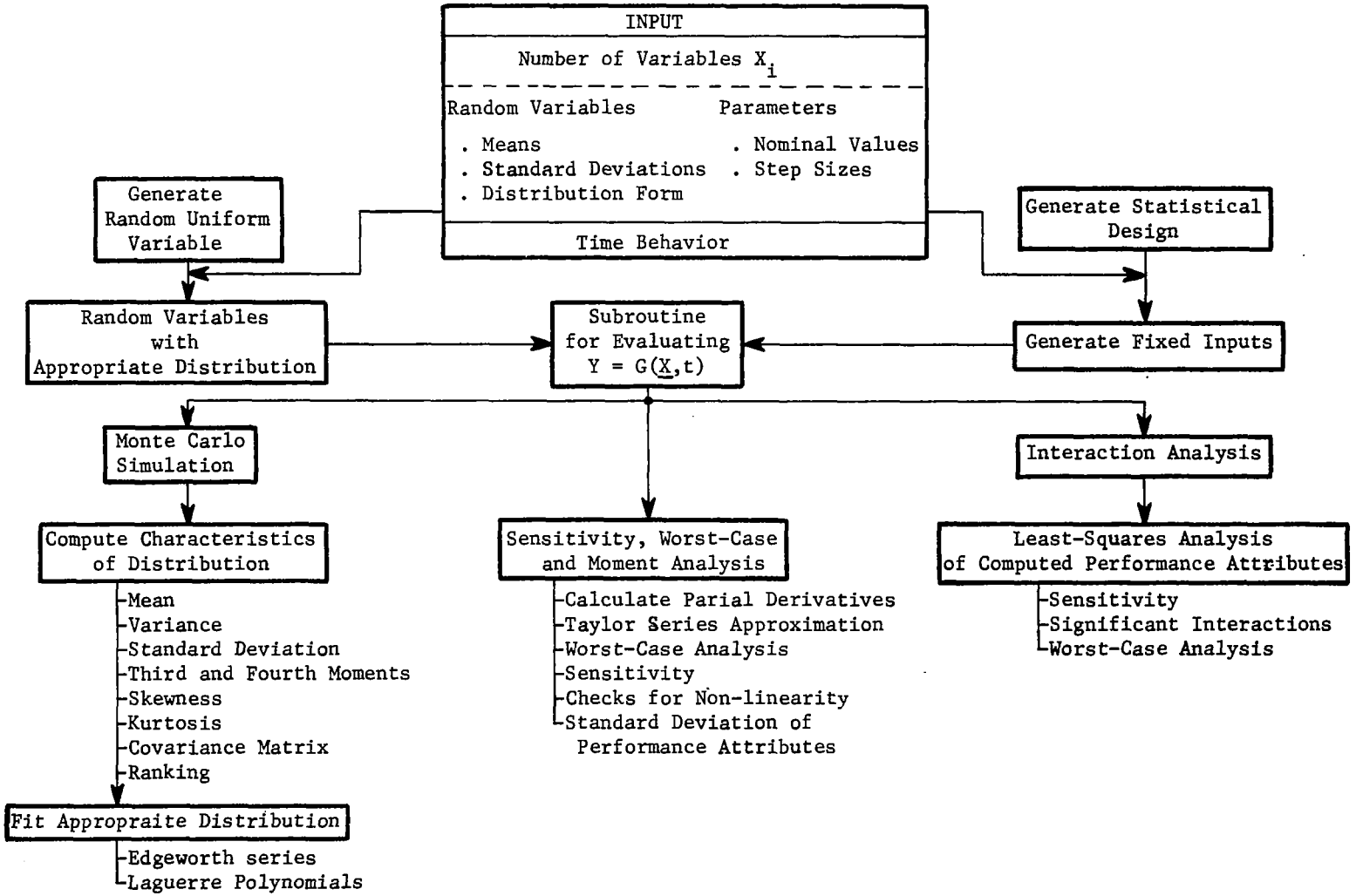


Figure 4-4. Flow Diagram for General PVA Program

(6) Gamma (Integral values of one parameter),

(7) Chi-Square,

(8) Triangular, and

(9) Beta (Integral values of both parameters).

Uniformly distributed variables are first generated; they are then transformed according to the methods described in Ref. 4-3, Appendix B to variables having the appropriate distributions as specified in the input. These transformed variables are then used to compute the performance measures such as voltage output, current output, power dissipation,etc. The performance measures are generated the number of times required to obtain the desired precision of the results. When the inputs are precisely known, the number of trails necessary to estimate the distribution function of a performance measure to the required degree of precision for a one-dimensional distribution can be estimated from the Kolmogoroff-Smirnov statistic for the maximum deviation d between the sampled distribution function and the true (but unknown) distribution function. Table 4-2 displays the number of observations necessary in order that the probability be α that the maximum deviation between the distribution function and the sample function exceeds the value d.

Table 4-2
Percentiles of the Distribution of d
for Several Values of $1-\alpha$

	$1-\alpha$				
N	0.80	0.85	0.90	0.95	0.99
5	0.45	0.47	0.51	0.56	0.67
10	0.32	0.34	0.37	0.41	0.49
20	0.23	0.25	0.26	0.29	0.35
30	0.19	0.20	0.22	0.24	0.29
40	0.17	0.18	0.19	0.21	0.25
50	0.15	0.16	0.17	0.19	0.23
For larger values of N	$\frac{1.07}{\sqrt{N}}$	$\frac{1.14}{\sqrt{N}}$	$\frac{1.22}{\sqrt{N}}$	$\frac{1.36}{\sqrt{N}}$	$\frac{1.63}{\sqrt{N}}$

Hence, if N is 50 the chance is 0.05 that the maximum deviation between the sample distribution function and the actual distribution function exceeds 0.19; if N = 100, d = 0.136, and if N = 1000, d = 0.043. When high precision is needed, it is possible to perform a very large number of simulation trials. However, it must be remembered that the cost in computer time per simulation depends on the complexity of the performance model subroutine.

In practice the distributions of the component characteristics are seldom known very precisely. Hence there is a precision of the distribution of the performance measure beyond which it is impractical to attemp to refine the estimate of the true distribution. In fact, very often a uniform distribution of the input variable is assume because of the lack of knowledge concerning the true distribution.

Suppose now that a rational procedure is available for estimating N and that N values of the performances have been computed. Then the N observations are ranked in ascending order of performance, their first four central moments are computed, and the measures of skewness and kurtosis are obtained. From the statistics it can be decided which distribution to fit to the data or which series approximations to use. The approximating distributions can be fitted by the method of moments.

In this program the Edgeworth series and/or Laguerre polymonials are used to approximate the unknown distribution function. The methods for fitting these distributions are given by Kendall [Ref. 4-12].

Sensitivity and Moment Analysis

This program obtains Taylor series approximation to the models and as illustrated in Fig. 4-4 subsequently uses them to predict worst-case performances, to estimate sensitivities of performance measures to inputs, to check for nonlinearities and interactions of behavior with respect to inputs, and to perform a moment analysis. The step sizes are chosen to include the expected range of variation of the input variables as a result of the environments described by the mission profile, the inherent variations in the part characteristics, and the aging effects.

This part of the program first computes estimates of the first and second partial derivatives of the performance measures of interest with respect to each of the pertinent part characteristics, inputs, loads, etc.; the five-point central difference formulas are used for obtaining the partial derivatives.

Having obtained the first and second partial derivatives of a performance measure with respect to the independent variables, the following Taylor series expansion is obtained.

$$Y(h_1,h_2,\ldots,h_m) \simeq Y_N + \Sigma Y_i' h_i + \frac{1}{2} \Sigma Y_i'' h_i^2 + \ldots \qquad (4\text{-}11)$$

where

Y_N is the nominal value of performance measure Y,

Y_i'&Y_i'' are respectively the 1-st and 2-nd partials of Y with respect to input variable X_i,

h_i is the change from nominal of X_i, and the sums are over all m input variables.

Dividing by Y_N yields

$$\frac{Y}{Y_n} \simeq 1 + \Sigma LS_i + \Sigma QS_i \ , \tag{4-12}$$

where

LS_i = a measure of linear sensitivity of the performance measure to the i-th input variable

$$LS_i = \frac{Y_i' h_i}{Y_N} \tag{4-13}$$

and

QS_i = a measure of second degree or quadratic sensitivity (denoted as nonlinear sensitivity in the program output) of the performance with respect to the i-th input variable and is given by

$$QS_i = \frac{1}{2} Y_i'' h_i^2 / Y_N. \tag{4-14}$$

These two quantities are printed out for each of the N variables. The sensitivity measure associated with the i-th variable is essentially the relative change in the performance measure as a function of the maximum expected change in the i-th variable. The definitions of sensitivity and non-linearity are suggested by the Taylor series expansion. As noted earlier, there are several definitions of sensitivity appearing in the literature. The definitions used in this program are very convenient in estimating the relative change in a performance measure Y for the expected changes in the independent variables.

The Taylor series expansion as presented above does not include terms with mixed partial derivatives. To obtain the second partial derivatives with respect to all pairs of independent variables would require considerably more computing time. The computation is performed using only the first partials and the pure second partials; the series approximation is then checked for its adequacy. If the results are not as precise as required, the appropriate mixed second partials are obtained by a program described in the section on Interaction Analysis.

Worst-Case Limits

The worst-case limits are computed by the procedure described by West and Scheffler [Ref.4-13]. The signs of the first partial derivatives are examined; the variables for which they are positive are placed at their high values, X + h, and the variables for which they are negative, at their low values, X - h , in order to estimate an upper worst-case limit. Conversely, to estimate a lower limit the variables for which Y' is positive are placed at their low values, and for Y' negative, at their high values. The worst-case limits of the performance measures are computed by actually substituting the appropriate values of the variables into the functions comprising the performance model. The computed worst-case limits are then compared to the limits estimated with the Taylor series expansion. If these values do not agree to within the required accuracy, the omitted terms, namely, the mixed partial derivatives (interactions) and the higher order pure terms must be investigated. The higher order pure derivatives are conveniently checked one variable at a time by comparing the functional value at the two end points with that estimated by the first and second partials with respect to that variable. These checks suggest the source of any lack of precision.

Moment Analysis

The moments of the performance measures can be obtained from the Monte Carlo simulation runs or from an error propagation analysis based on the Taylor series approximation. The latter is simpler to compute and not subject to sampling fluctuations as is the former. However, the series approximation is subject to the lack of precision with which it approximates the true function.

Let

$$Y \simeq Y_n + \sum_i \frac{\partial Y}{\partial X_i}\bigg|_{\underline{X}_N} \Delta X_i + \frac{1}{2}\sum \frac{\partial^2 Y}{\partial X_i^2}\bigg|_{\underline{X}_N} \Delta X_i^2$$

$$+ \frac{1}{2}\sum\sum \frac{\partial^2 Y}{\partial X_i \partial X_j}\bigg|_{\underline{X}_N} \Delta X_i \Delta X_j \ .$$

If only the first order terms are used, the estimates of the mean and variance of Y, denoted by $\hat{\mu}\{Y\}$ and $\hat{\sigma}^2\{Y\}$ respectively, are given by

$$\hat{\mu}\{Y\} = Y_N$$

$$\hat{\sigma}^2 Y = \sum\sum \frac{\partial Y}{\partial X_i}\bigg|_{\underline{X}_N} \frac{\partial Y}{\partial X_j}\bigg|_{\underline{X}_N} \hat{Cov}\{X_i, X_j\}$$

where

$\hat{Cov}\{X_i,X_j\} = \hat{\sigma}\{X_i\}\ \hat{\sigma}\{X_j\}\ \rho\{X_i,X_j\}$

$\hat{\sigma}\{X_i\}$ = estimated standard deviation of the measurements X_i,

$\rho\{X_i,X_j\}$ = estimated simple correlation of the measurements on X_i and X_j.

If X_i and X_j are characteristics of two distinct components, then $\rho\{X_i,X_j\} = 0$; otherwise, it is estimated by

$$\rho\{X_i,X_j\} = \frac{\Sigma(X_{ik}-\bar{X}_i)(X_{jk}-\bar{X}_j)}{\{\Sigma(X_{ij}-\bar{X}_i)^2 \cdot \Sigma(X_{jk}-\bar{X}_j)^2\}^{1/2}} .$$

If the first and second order terms (not including the mixed partials-interactions terms) are used in the approximation, then further terms are required in the moment analysis.

Let

$$Y_i' \quad \text{denote} \quad \left.\frac{\partial Y}{\partial X_i}\right|_{\underline{X}_N}$$

$$Y_{ij}'' \quad \text{denote} \quad \left.\frac{\partial^2 Y}{\partial X_i \partial X_j}\right|_{\underline{X}_N} ,$$

then the estimated mean and variance for Y can be written as

$$\mu\{Y\} = Y_N + \frac{1}{2}\Sigma Y_i''^2\ \hat{\sigma}^2\{X_i\}$$

$$\hat{\sigma}^2\{Y\} = \Sigma Y_i'^2\ \sigma^2\{X_i\} + \frac{1}{4}\Sigma Y_i''^2 [\hat{\mu}_{4i} - \hat{\sigma}^4\{X_i\}]$$

$$+ \Sigma\Sigma Y_i' Y_j'\ \hat{Cov}\{X_i,X_j\}$$

$$+ \frac{1}{4}\Sigma\Sigma Y_i'' Y_j''\ [\hat{E}\{\Delta X_i^2\ \Delta X_j^2\} - \hat{\sigma}^2\{X_i\}\ \hat{\sigma}^2\{X_j\}]$$

$$+ \frac{1}{2}\Sigma Y_i' Y_i''\ [\hat{\mu}_{3i}]$$

$$+ \frac{1}{2}\Sigma\Sigma Y_i' Y_j''\ \hat{E}\{\Delta X_i\ \Delta X_j^2\} ,$$

where E{X} denotes the expected or mean value of X and $\hat{\mu}_{3i}$ and $\hat{\mu}_{4i}$ are the estimated third and fourth moments of X_i, i=1,...,m. A similar expansion may be obtained with the interaction terms included.

In the above analysis it has implicitly been assumed that the relationship between the performance measure Y and the part characteristics, X_i, i=1,...,m is known, that is, the coefficients are known. However, in practice the relationship may be obtained from empirical data and the coefficients may be considered estimates of true but unknown values. The extent to which the data are available should then be reflected in the precisions of the inputs to the error propagation analysis. A complete discussion of this problem is given in Marini, Brown, and Williams [Ref. 4-14].

Interaction Analysis

In case the worst-case limits computed directly from the functions are not adequately approximated by the linear and pure quadratic terms, it is necessary to compute the mixed partial derivatives for the pairs of variables which are expected to yield significant interaction effects. The mixed partials can be computed by one of the following two methods.

One procedure would be to compute the first partial derivatives with respect to the i-th variable at five different values of the j-th variable. These partials would in turn be used to compute the second partial. This procedure assumes a degree of smoothness of the analytical function.

A second procedure would be to generate the performance measure for selected sets of values of the independent variables and then fit by regression techniques the functional form

$$Y = b_0 + \Sigma\, b_i X_i + \Sigma\, b_{ii} X_i^2 + \Sigma\Sigma\, b_{ij} X_i X_j \ .$$

This assumes all higher order effects can be adequately accounted for by a second degree polynomial function. The coefficients of the terms $X_i X_j$ would correspond to the mixed partials under the assumption. The selection of the values of the variables can be performed efficiently by the method of statistical designs for factorial experiments. Methods for generating the appropriate design are described by Addelman [Ref. 4-15].

4.3.1.1 General PVA Program Example

The general PVA program which has been discussed in the preceding pages can be used to perform a wide variety of variation analyses for a wide variety of systems. Some examples using this program are given in Ref. 4-3. A simple example is reproduced here from that reference.

A second degree polynomial was chosen for illustration of the program.

$$Y = 1 + 2X_1 + 2X_2 + 3X_1X_2 + 4X_1^2 + 4X_2^2 .$$

There are two independent variables, X_1 and X_2, and one dependent variable Y denoted by POLY in the program input. One hundred (100) simulation trials were performed assuming X_1 and X_2 are normally distributed with means 10 and 5 and standard derivations 0.2 and 0.05, respectively, and correlation 0.5.

In the interaction analysis part of the program, one needs to indicate which independent variables, from those available, are to be used in the analysis. In the specific example there are only two such variables and both of them are used as indicated by inputs 4 and 5. If there were 10 variables in all and only five variables to be used in the analysis, e. g. variables numbered 1, 3, 5, 8, and 10, then input 5 would be these numbers in the appropriate format and input 4 would be NVT = 5 and NVU = 5 provided all 2^5 combinations of the 5 variables were used. See Addelman [Ref.4-15] for methods of statistical design of experiments for using a fraction of 2^5 runs. The inputs and outputs for various parts of the program are listed on the following pages. The program outputs are from the Bunker-Ramo 340 computer; the program is written in FORTRAN II language. For convenience of reproduction of this report, the printout from the program has been reproduced by typing. The printout format has been preserved.

Program Input Description for Simulation

(1) The first card has the starting value, XN, for the random number generator. Format (F10.0).

(2) Input card 2 gives the number of models (not more than five) followed by a four letter identifier for each model. Format (I2,5A4).

(3) This card provides the actual number of variables and the number of correlated variables for each model, and the number of simulation trials for all models. Format (11I5).

(4) These cards contain information necessary for a readable output. The first contains the names of the distributions of the random number generators (each name is limited to twelve characters). The second has the names of the two polynomial fit routines,namely Edgeworth and Laguerre. Format (20A4).

(5) The variable input cards contain nominal and deviation values, a parameter name, and a random number generator call value. The call value is the argument for a COMPUTED GO TO statement and calls the appropriate generator subroutine. Format (2E10.4,A4,I4). Those variables which have non-zero correlations with other variables must be read in first.

(6) If there are correlated variables, the values are read as an upper triangular matrix. Format (16F5.0).

Input Description for Sensitivity,Worst-Case, and Moment Analysis

(1) Model identification is on the first card. The number of models, not to exceed 10, is followed by four letter model descriptors. Format (I2,10A4).

(2) The next card gives the variable information for each model. The number of variables for each model, not to exceed 20, is in Format (10I2).

(3&4) These cards are identical to the simulation input card types (5) and (6). The nominal and deviation values (one-half the expected extreme deviation values) are in the same format and the variable name should also be given, (2E10.4,A4). Correlated variables, Format (16F5.0).

Input Description for Interaction Analysis

(1) Card one is for the number of models, Format (I2).

(2) Card two specifies the total number of independent variables (NV) and the (alphanumeric) name for the dependent variable. Format (I2,A4).

(3) The variable cards specify the nominal values and deviations of each independent variable, as well as its (alphanumeric) name. There is one card for each variable. Format (2E10.4,A4).

(4) This control card indicates the number of variables (NVT) to be used in the interaction analysis (NVT $\leq$ NV) and the number of variables whose levels are to be computed (NVU). If NVT = NVU, all combinations are considered; otherwise NVU < NVT. Format (2I2).

(5) Card five indicates, by subscripts, the variables selected for analysis. The number of values appearing should be NVT in format (20I2).

(6) Card 6 is omitted if NVT = NVU. Otherwise it specifies, by subscripts, the NVU variables to be computed. Format (20I2).

Cards 2-6 are repeated for each model. The deviations specified on Card 3 are doubled for the least squares analysis. That is, the upper and lower limits considered for each variable are the nominal values plus and minus twice the deviations given on Card 3.

The program inputs to the example using the polynomial introduced as the performance model are given in Table 4-2 and followed by the outputs in Tables 4-3 through 4-5.

Table 4-2

Program Inputs for Polynomial (POLY) Example

Inputs (Card Image)

Simulation Analysis

```
(1)          1697.
(2)      1POLY
(3)         2     2 100
(4)     UNIFORM     NORMAL       LOG NORMAL  EXPONENTIAL WEIBULL      GAMMA      BETA
            CHI SQUARE
        EDGEWORTH   LAGUERRE
(5)      .1000E 02 .2000E 00  X1       2
         .5000E 01 .5000E-01  X2       2
(6)      1.0  0.5  1.0
```

Sensitivity, Worst-Case and Moment Analysis

```
(1)      1POLY
(2)      2
(3)      .1000E 02 .2000E 00  X1       2
         .5000E 01 .5000E-01  X2       2
(4)      1.0 0.5   1.0
```

Interaction Analysis

```
(1)      1
(2)      2POLY
(3)      .1000E 02 .2000E 00  X1
         .5000E 01 .5000E-01  X2
(4)      2 2
(5)      1 2
```

Table 4-3

Simulation Output for POLY

MODEL 1, POLY	VAR. NAMES	NOMINAL VALUE	DEVIATION	DISTRIBUTION
	X1	.10000E 2	.200000E 0	NORMAL
	X2	.50000E 1	.500000E -1	NORMAL

INPUT CORRELATIONS

.500

INPUT CHECK

MODEL 1, POLY	VAR. NAMES	NOMINAL VALUE	DEVIATION	DISTRIBUTION
	X1	.99866E 1	.20871E 0	NORMAL
	X2	.50019E 1	.65881E -1	NORMAL

INPUT CORRELATIONS

.608

DEPENDENT DATA LISTED IN ASCENDING ORDER

I	I/N	POLY							
1	.010	.6322E 3	39	.390	.6735E 3	77	.770	.6962E 3	
2	.020	.6346E 3	40	.400	.6740E 3	78	.780	.6981E 3	
3	.030	.6362E 3	41	.410	.6740E 3	79	.790	.6983E 3	
4	.040	.6389E 3	42	.420	.6748E 3	80	.800	.6986E 3	
5	.050	.6403E 3	43	.430	.6750E 3	81	.810	.7012E 3	
6	.060	.6434E 3	44	.440	.6755E 3	82	.820	.7013E 3	
7	.070	.6478E 3	45	.450	.6758E 3	83	.830	.7021E 3	
8	.080	.6494E 3	46	.460	.6764E 3	84	.840	.7024E 3	
9	.090	.6504E 3	47	.470	.6774E 3	85	.850	.7039E 3	
10	.100	.6516E 3	48	.480	.6786E 3	86	.860	.7040E 3	
11	.110	.6554E 3	49	.490	.6786E 3	87	.870	.7051E 3	
12	.120	.6567E 3	50	.500	.6788E 3	88	.880	.7056E 3	
13	.130	.6579E 3	51	.510	.6790E 3	89	.890	.7062E 3	
14	.140	.6580E 3	52	.520	.6791E 3	90	.900	.7067E 3	
15	.150	.6594E 3	53	.530	.6799E 3	91	.910	.7096E 3	

Table 4-3 (Continued)

16	.160	.6607E 3	54	.540	.6817E 3	92	.920	.7099E	3
17	.170	.6612E 3	55	.550	.6820E 3	93	.930	.7113E	3
18	.180	.6614E 3	56	.560	.6833E 3	94	.940	.7117E	3
19	.190	.6615E 3	57	.570	.6851E 3	95	.950	.7125E	3
20	.200	.6615E 3	58	.580	.6856E 3	96	.960	.7147E	3
21	.210	.6634E 3	59	.590	.6858E 3	97	.970	.7158E	3
22	.220	.6637E 3	60	.600	.6858E 3	98	.980	.7167E	3
23	.230	.6647E 3	61	.610	.6860E 3	99	.990	.7272E	3
24	.240	.6650E 3	62	.620	.6860E 3	100	1.000	.7288E	3
25	.250	.6659E 3	63	.630	.6863E 3				
26	.260	.6660E 3	64	.640	.6864E 3				
27	.270	.6661E 3	65	.650	.6872E 3				
28	.280	.6676E 3	66	.660	.6882E 3				
29	.290	.6678E 3	67	.670	.6892E 3				
30	.300	.6692E 3	68	.680	.6893E 3				
31	.310	.6693E 3	69	.690	.6893E 3				
32	.320	.6696E 3	70	.700	.6903E 3				
33	.330	.6704E 3	71	.710	.6904E 3				
34	.340	.6706E 3	72	.720	.6906E 3				
35	.350	.6711E 3	73	.730	.6933E 3				
36	.360	.6715E 3	74	.740	.6933E 3				
37	.370	.6726E 3	75	.750	.6951E 3				
38	.380	.6734E 3	76	.760	.6953E 3				

MOMENTS	POLY	PERCENTAGE POINTS FOR POLY BY EDGEWORTH	
FIRST	.680057E 3	Z = 616.96093	F(Z) = -.82690E -2
SECOND	.437902E 5	Z = 627.47654	F(Z) = -.11345E -1
THIRD	-.243477E 5	Z = 637.99217	F(Z) = .88139E -2
FOURTH	.503237E 8	Z = 648.50779	F(Z) = .83527E -1
STD. DEV.	.210316E 2	Z = 659.02342	F(Z) = .21687E 0
SKEWNESS	-.265701E -1	Z = 669.53905	F(Z) = .36661E 0
KURTOSIS	.262433E -1	Z = 680.05467	F(Z) = .49822E 0
VARIANCE - COVARIANCE MATRIX, ORDER 1		Z = 690.57030	F(Z) = .63100E 0

Table 4-3 (Continued)

POLY .468040 E3

Z = 701.98592	F(Z) =	.78309E	0
Z = 711.60155	F(Z) =	.91788E	0
Z = 722.11718	F(Z) =	.99259E	0
Z = 732.63280	F(Z) =	.10121E	1
Z = 743.14842	F(Z) =	.10086E	1

Table 4-4

Sensitivity, Worst-Case, and Moment Analysis Output for POLY

FIRST AND SECOND PARTIAL DERIVATIVES (Y' AND Y'') OF POLY WITH RESPECT TO X

					PARTIALS		SENSITIVITY	
X	Y(X-2DX)	Y(X-1DX)	Y(X+1DX)	Y(X+2DX)	Y'	Y''	LINEAR	NON-LIN
X1	.64284E 3	.66175E 3	.70055E 3	.72043E 3	.96986E 2	.79590E 1	.56967E -1	.93499E -3
X2	.67384E 3	.67740E 3	.68460E 3	.68823E 3	.71995E 2	.78125E 1	.10572E -1	.57361E -4

ALL X AT NOMINAL, Y(X) = .68099E 3

STD DEV OF Y(X), .21425E 2

WORST CASE LIMITS

VALUE OF VARIABLE AT LOWER LIMIT AND AT UPPER LIMIT ,		X	DX
X1	.96000E 1	.10400E 2	.10000E 2	.20000E 0
X2	.49000E 1	.51000E 1	.50000E 1	.50000E -1

WORST CASE LIMITS AND NOMINAL VALUE

POLY	.63579E 3	.72779E 3	.68099E 3

INTERACTION CHECK USING 1ST AND 2ND DEGREE TERMS OF TAYLOR SERIES

POLY	.63567E 3	.72766E 3

INTERACTION CHECK USING 1ST DEGREE TERMS OF TAYLOR SERIES

POLY	.63500E 3	.72698E 3

GOODNESS OF FIT USING 1ST AND 2ND TERMS OF TAYLOR SERIES

VARIABLES	Y(X-2DX)/Y(X)	1.-SENS	1.-SENS+NON LIN	Y(X+2DX)/Y(X)	1.+SENS	1.+SENS+NON LIN
X1	.94397E 0	.94303E 0	.94397E 0	.10579E 1	.10570E 1	.10579E 1
X2	.98949E 0	.98943E 0	.98948E 0	.10106E 1	.10106E 1	.10106E 1

Table 4-5

Interaction Analysis Output for POLY

```
VARIABLE       NOMINAL VALUE          DX

 X1             .10000E  2      .20000E  0
 X2             .50000E  1      .50000E -1

CODED LEVELS OF THE VARIABLES X(I)
    0-LOW LEVEL      1-HIGH LEVEL

ROW              MOD-2 ARRAY OF VARIABLES

 1       0    0
 2       0    1
 3       1    0
 4       1    1

ACTUAL LEVELS OF X(I) AND CORRESPONDING PERFORMANCE VALUES

ROW          X1             X2             POLY

 1    .96000E  1     .49000E  1     .63579E  3
 2    .96000E  1     .51000E  1     .64995E  3
 3    .10400E  2     .49000E  1     .71316E  3
 4    .10400E  2     .51000E  1     .72779E  3

COEFFICIENTS OF VARIABLES AND THEIR SENSITIVITIES

                                    COEFFICIENTS       SENSITIVITY

         CONSTANT          B(  0) =    .68167E  3
            X1             B(  1) =    .96938E  2        .56938E -1
            X2             B(  2) =    .71480E  2        .10496E -1
      X1  , X2         B(  1,  2) =      .29087E  1         .85423E  -4
```

4.3.2 ECAP and NASAP for PVA

The Electronic Circuit Analysis Program (ECAP) was developed jointly by IBM and Norden Division of United Aircraft; Ref.4-16 is the basic reference for the program. ECAP is very widely used for circuit analysis; it is available from IBM for use on the IBM 1620, 7000 series, and 360 series computers, although not all of these versions are officially supported by IBM [Ref.4-17]. It has been suitably modified by other organizations for use on a variety of other computers and with some valuable additional features for PVA.

In the versions of ECAP available from IBM, the PVA capabilities include the following [Ref.4-18]:

For dc analysis:

(1) partial derivative of voltage at a particular circuit node with respect to a circuit parameter in a particular branch;

(2) sensitivity of a node voltage with respect to a branch parameter;

(3) worst-case solutions;

(4) standard deviation of circuit output variables;

(5) automatic parameter variation, which allows a parameter to be incremented over a range of values with a circuit solution computed for each value.

For ac analysis:

(1) automatic parameter variation.

Additional PVA capabilities which have been incorporated in ECAP by other organizations include ac sensitivities and solution of the propagation-of-variance equation [Ref.4-17].

The Network Analysis for System Application Program (NASAP) has been developed by NASA/Electronics Research Center in a cooperative effort involving about 20 users of the program [Ref.4-19]. NASAP is unique among circuit analysis programs in that it uses flowgraph techniques to analyze networks instead of matrix-oriented techniques. Also, it manipulates circuit symbolic parameters instead of actual parameters until the final step of the analysis. This symbol-manipulation feature has some interesting ramifications, among which are the ability to calculate partial derivatives and sensitivities symbolically [Ref.4-20].

In addition to the PVA capabilities noted above, NASAP incorporates an optimization procedure which eliminates from a circuit input parameters having less than a preassigned amount of influence on circuit performance parameters; the procedure is in effect a tolerance analysis [Ref.4-20].

NASAP was originally written in FORTRAN IV for use on the CDC 3600 computer; it also is now in use on several other computers. Although reportedly available from COSMIC [Ref.4-21], it does not appear in the July 1967 listing of COSMIC

programs [Ref.4-22]. However, it can be obtained [Ref.4-23] by contacting:

R. M. Carpenter
NASA/ERC
575 Technology Square
Cambridge, Mass.
Tel. 617 491-1500, Ext. 541

References

4-1. Milne, W.E.: Numerical Calculus. Princeton University Press, Princeton, N. J., 1949.

4-2. Staff of Autonetics Div. of North American Aviation: Reliability Analysis of Electronic Circuits. (Available from DDC as AD 461 303).

4-3. Nelson, A. C.; et. al.: Development of Reliability Methodology for Systems Engineering, Vols. I and II. Research Triangle Institute, Research Triangle Park, North Carolina, (NASA Contract NASw-905), April 1966.

4-4. Abramowitz, M.; and Stegun, I. A.: Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. Dover Publications, New York, New York, 1965.

4-5. Dorf, Richard C.: Computer-Analyze Your Circuit.. Electronic Design, Vol. 13, June 21, 1967, pp. 54-57.

4-6. O'Bryant, R. O.: Variability Prediction--A New Method. Proceedings 1967 Symposium on Reliability, Washington, D. C., pp. 181-188.

4-7 Freund, J. E.: Mathematical Statistics. Prentice-Hall, Inc., Englewood Cliffs, N. J., 1962.

4-8. Kiefer, F. P., et. al.: Final Report on Prediction of Circuit Drift Malfunction of Satellite Systems, Report ARPA 168-61. IBM, FSD Space Guidance Center, Owego, New York for Rome Air Development Center, Griffiss Air Force Base, Contract AF 30(602)-2418, AD 276 044, 1961.

4-9. Hayward, R. A.; and Ingber, J. I.: Monte Carlo Flight Performance Reserve Program. Report GD/C-BTD-65-176. General Dynamics Convair Division, STAR N66-22903, 1966.

4-10. Markowitz, H. M.; et. al.: Launch Vehicle Optimization - Phase II Final Report, Vol. II - Techniques Development. Lear Siegler, Inc., Contract NASw-398. 1965, STAR N66-14951.

4-11. Bosinoff, I.; et. al.: Transfer Functions in Mathematical Simulation for Reliability Prediction, Final Report, Report RADC-TDR-63-87, Sylvania Electronic Systems Command, Contract AF 30(602)-2376, 1963.

4-12. Kendall, M. G.: The Advanced Theory of Statistics, Vol. 1, Charles Griffin and Co., Ltd., London, England, 1948.

References (Continued)

4-13. West, W. J.; and Scheffler, H. S.: Design considerations for reliable electronic equipment. Northeast Research and Engineering Meeting, Boston, Mass., 1961.

4-14. Marini; Brown; and Williams: Evaluation and prediction of circuit performance by statistical techniques. Aeronautical Radio, Inc., Monograph No. 5, Publication No. 113.

4-15. Addelman, S.: Techniques for constructing fractional replicate plans, J. Am. Statis. Assoc., 58, 45-71.

4-16. IBM Tech. Publications Dept.: 1620 Electronic Circuit Analysis Program (ECAP) (1620-EE-02X). IBM Publication H20-0170-1, White Plains, N. Y., 1965.

4-17. Wall, H. M.: ECAP-1966. NEREM Record-1966, pp. 84-85.

4-18. Tyson , H. N., Jr.; Hogsett, G. R.; and Nisewanger, D. A.: The IBM Electronic Circuit Analysis Program (ECAP). Proceedings 1966 Annual Symposium on Reliability, pp. 45-65.

4-19. Carpenter, R. M.: NASAP--Network Analysis for System Application Program--Present Capabilities of a Maintained Program. Computer-Aided Circuit Design Seminar Proceedings, April 11-12, 1967, Kresge Auditorium, MIT, pp. 83-94.

4-20. Carpenter, R. M.; and Happ, W.: Computer-Aided Design--Part 3 Analyzing Circuits with Symbols. Electronics, Dec. 12, 1966, pp. 92-98.

4-21. Carpenter, R. M.: Topological Analysis of Active Networks. Proceedings of the Institute on Modern Solid State Circuit Design, Univ. of Santa Clara, California, Sept 1966, pp. 50-59.

4-22. Computer Center, Univ. of Georgia: A Directory of Computer Programs Available from COSMIC, Vol. 1, July 1, 1967.

4-23. Dumanian, J. A., ed.: IEEE CADAR Newsletter. January 1967.

5. Part Application Analysis

In part application analyses the operating stresses of the individual components are determined and compared to the rated capabilities. In an electronic circuit, for example, part stresses such as power dissipation of a resistor, peak reverse voltage of a diode, and voltage across a capacitor are all tabulated and compared to their electrical ratings. The concept of stress here is an extension of the concept of mechanical stress applied in strength of materials analysis and is broadened to include electrical, thermal, radiation and other potentially damaging effects that may jeopardize the acceptable operation of a component. The purpose of the analysis is to insure that actual component loads do not exceed the manufacturer's rated or user's derated capability of the component.

The significant application of computers in part application analysis is indirectly through other types of analyses such as circuit, thermal and structural analyses. For example, with circuit analysis programs such as ECAP, node voltages and branch currents (hence branch component power dissipation) of electrical networks can be computed for later comparison to rated conditions. The circuit analysis program NET-1 allows as input the rated dc conditions of certain components, performs a comparison against rated values as a part of the analysis, and prints out an alarm if a computed parameter value exceeds the input rated value. Mechanical stress analysis is usually an inherent feature in structural analysis programs, since the stress level in a structure is concerned with the primary function of the structure.

The computer can serve as an aid to application analyses on system components for any situation in which the component loads can be computed with an appropriate model. Vol. V of this report series treats part application analyses in some detail.

6. Failure Mode and Effects Analysis (FMEA)

This analysis task is approached in several ways. The common purpose of all approaches is to determine what discrepancies can occur in a system, identify their effects on system operation, and eliminate those that are more critical and more likely to occur. A large portion of the analysis relies on engineering judgement and is thus performed manually. Computers can assist, but the extent of applicability depends on the approach taken and the nature of the system. FMEA remains the important procedure for actually uncovering the system discrepancies. It is in fact one of the most important activities in the total design for reliability process since it identifies areas requiring action by other design activities. One of its important outputs is the designation of the logic models for individual elements to be included in reliability prediction calculations.

One of the simplest approaches to FMEA is: given a design configuration, each of the components and materials comprising the design can fail or degrade via a number of different modes. The failure mode analysis consists of nothing more than explicitly identifying these modes. For a system composed of discrete components, this identification involves merely proceeding through a parts list and deciding what modes of failure are to be considered. There is a practical limit, of course, as to how many failure modes of each part can be considered, and in fact a limited failure effects analysis is performed on a subjective basis at this stage to aid in limiting the number of modes considered.

For electronic circuits it is becoming fairly common to consider at least shorts and opens between all terminal pairs of components. Typical modes define the extreme discrete states of the components. It is possible to define in-between states, such as discrete levels of resistance for a resistor which differ from nominal, but the resulting analysis can quickly become unweildy if carried too far, especially when considering devices as complex as a transistor.

When for FMEA the lowest level of breakdown is limited to complex subassemblies (such as transmitters, power inverters, pumps, and engines) the failure modes become much more difficult to define. If these subassemblies are required to perform in sequences of operations, failure modes of the following types may be identified:

(1) premature operation,

(2) failure to operate at a prescribed time,

(3) failure to cease operation at a prescribed time, and

(4) failure during operation.

Within each of these modes there may be further modes to consider. For example, failure of a power supply during operation may be evidenced by either no output

voltage, loss of voltage regulation, frequency out of tolerance, or excessive voltage imbalance between different phases.

The only aid provided by a computer in the failure mode portion of the analysis is that of record keeping to eliminate manual drudgery. This role becomes more useful when the records can be used as input to the failure effects analysis, which potentially lends itself to more computer assistance.

The use of the computer in the failure effects portion of the analysis is primarily in the role of function evaluation using performance models to compute changes in performance due to particular failure modes. For example, considerations of fatigue failure of a particular structural member will not alter the basic form of the stiffness matrix but will modify the value of certain parameters. Upon substituting the modified values into the computer program for solving these structural equations, the computer can be used to evaluate the effect.

It is possible to extend certain performance evaluation programs to automatically perform these calculations for all failure modes to be investigated. The NET-I network analysis program [Ref. 6-1] does this upon input request for a limited number of abnormal modes of circuit voltage supplies and prints out the value of circuit performance parameters for each. NET-I does not automatically consider failure modes such as shorts and opens of circuit components; investigation of these would require manually setting up a new run to be made for each mode.

Most circuit analysis programs e.g., ECAP which accept a topological input description of the circuit and synthesize the circuit equations can be used to evaluate failure effects, but computer run time can become excessive since the circuit equations may have to be generated again for each run. Specifying an extreme failure mode such as an open or a short of a component essentially changes the circuit configuration and a completely new solution is required. A useful approximation to open or short failures often used is to maintain the same circuit configuration and merely use extremely high or low values of part parameters to simulate failures. For example, an extremely high capacitance value can effectively simulate a short of a capacitor for AC analysis but does not have the same effect on circuit equations as does a short.

The AMAP (Automated Failure Mode Analysis Program) circuit analysis program [Ref. 6-2] is one program which automates the failure effect analysis for dc circuits. It repeatedly solves the circuit equations, computing and printing circuit node voltages, for failure modes such as open and short for parts and shorts between all node pairs. As described in the reference, AMAP includes only resistors, diodes, transistors, power supplies and nodes. This automated approach to failure effects

analysis can carry over effectively in other types of systems such as structures and propulsion, but no programs are known which provide these capabilities.

As mentioned earlier, there is a practical limit to the number of failure modes of each component or material than can be considered, even with computers. As a result, most failure effects analyses are limited to first-order effects, i.e., to considering the effect of a single failure mode of one component at a time and ignoring combinations. The AMAP program does include second order effects to a limited extent, including open and short combinations between different terminal pairs of a transistor.

One of the major uses of the outputs of FMEA is identification of the models for individual elements to be used in a reliability prediction analysis. For example, the results can be used to decide whether a short or a particular resistor should be included in the prediction as a failure or successful operation. Another use is to aid in determining if there are any overstress conditions on circuit parts. The identification of failure effects also assists in compiling a failure dictionary to be used in fault diagnosis and test point allocation.

Another approach to FMEA is to apply the above procedure in reverse, i.e., to define a degraded or failed mode of the system and look for those component and material failures that can cause it. The approach is employed mainly for studying the mission sequences of functions for large systems. In this approach the excitation or "calling-up" of a function depends on the mode of operation of a function in a prior time interval. Typically there is only one path through the network for normal operation. Any other path corresponds to degradation or failure

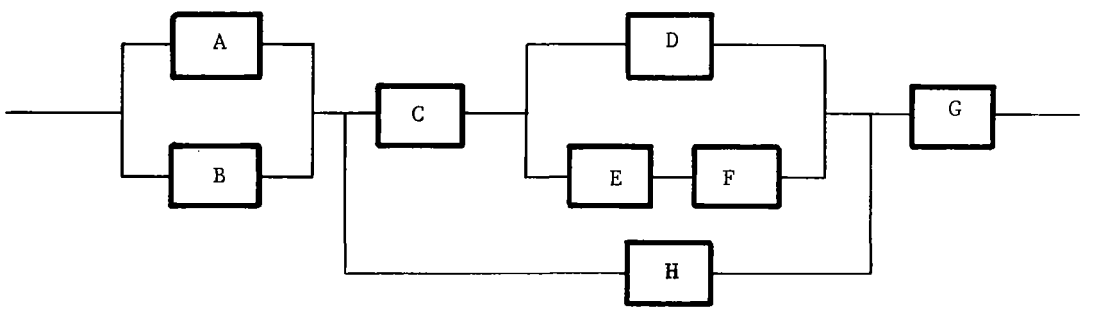


Figure 6-1. Fixed System with Redundancy

and will excite functions which cause the mission to be aborted, failed, or completed in a degraded mode. Thus given a particular outcome or terminating mode for the system, the analysis can search out those event combinations that can lead to it.

A third approach to FMEA is useful for systems with fixed configurations and containing extensive redundancy. Consider the conventional system logic diagram shown in Fig. 6-1. A first-order FMEA performed from this diagram is trivial since it was required prior to diagram construction anyway. A second-order FMEA shows that combinations such as elements A and B and elements D, F, and H cause system failure. When the redundancy gets very complex, the computer can assist in performing the higher-order FMEA.

References

6-1. Malmberg, A. F.: NET-1 Network Analysis Program. Proceedings 1965 Symposium on Reliability and Quality Control, pp. 510-517.

6-2. Hausrath, D. A.; and Ranalli, R.: Computer Studies of Abnormally Operating Circuits. Proceedings 1966 Annual Symposium on Reliability, pp. 66-86.

7. Reliability Prediction

A basic definition of the reliability of an equipment is the probability that the equipment successfully performs its intended function for a specified duration while operating under certain environmental conditions. Reliability prediction is the practice of using mathematical models to estimate this probability or related measures such as probability of failure, life distributions, or mean-time-to-failure. In addition to these estimates of system reliability alone, prediction of more complex measures of system worth related to reliability can be made. For example, it may be desired to optimize system reliability under cost constraints; a computer program which accomplishes this optimization is discussed later. Rarely are the models or statistics sufficient to obtain an estimate with sufficient accuracy to have meaning in the absolute sense. However, the results do frequently have meaning as a basis for selecting the best of several candidate designs, and the practice of predicting system reliability is now found in almost all system development programs.

Models and techniques for prediction are described in Vol. IV - Prediction of this series; we here emphasize the automation of the prediction analyses.

Reliability predictions are performed both on individual items and on the combinations of items forming higher levels of assembly up to and including the largest of systems. For individual items the analysis is usually so simple as to have no need for a computer. Computers do find considerable application in the analysis for combined items.

The common basis for all reliability predictions is the logic which defines the events of interest. This logic comprises the system model; not surprisingly it is called the prediction model. The event of the system being in a particular state (in simplest form the state is either success or failure) is the logic combination of other events associated with states of system subassemblies, inputs to the system, loads on the system, and/or system environmental conditions. In concept the logic comprising the preciction model can allow any number of different states of a part; most analyses of complex equipment employ simple two-state models (success vs. failure) to limit analysis complexity.

The basic flow of procedures in reliability prediction is shown in Fig. 7-1. A major milestone is the prediction model, from which either of two basic approaches may be followed. The approach illustrated by the upper path leads to a prediction equation which expresses the probability of system success or a related measure as a function of individual element probabilities. One of the simplest roles of computers in reliability prediction is to use such an equation programmed for estimating system probabilities and computing sensitivities of system probabilities to changes

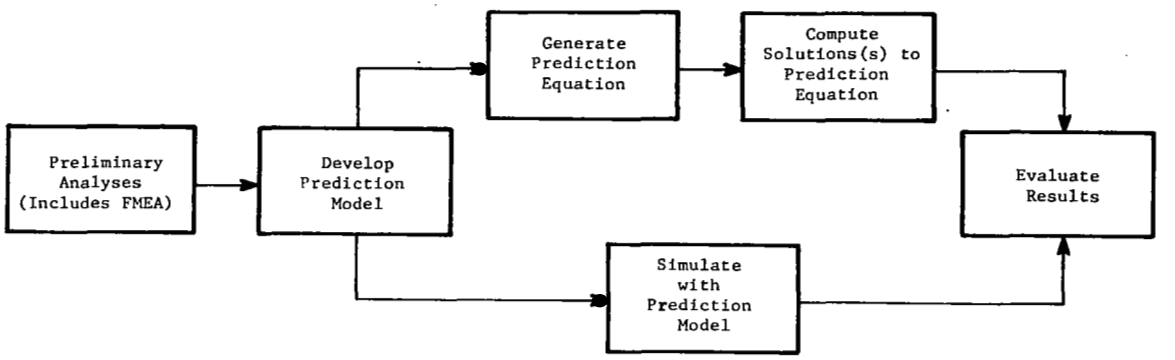


Figure 7-1. Reliability Prediction Process

in probabilities of subsystem events. This is especially appropriate when the prediction equation is derived manually and is too complex for manual solution. A computer application which implements the lower path in the figure is the use of the prediction model for simulating the system by Monte Carlo methods to estimate the probability of system success or other reliability parameters. Computers also can be used to cover various combinations of the steps illustrated in the figure.

7.1 Developing the Prediction Model

A prerequisite to the prediction analysis is a preliminary analysis of the equipment and its operational profile to establish mission functions, operating times and sequences, and environments. A failure modes and effects analysis as described earlier is an important part of the preliminary analysis, particularly for complex systems. An output of the FMEA is identification of the logic models to be used to single elements in the model for the prediction analysis, which is next established. The goal is to obtain a logic representation which relates reliability events of interest (such as system success) to the events that cause them. This logic can be developed in two principal ways as described below.

When a system is a fixed configuration or when the event of system success during a particular phase of system operation is concerned only with the fixed system

configuration which exists during this phase, a logic diagram is constructed which typically reveals the various logic elements operating in series or parallel. This diagram usually is derived manually from functional diagrams, schematics, special analyses, and general knowledge of system operation. Forms of logic relationships that can be used in prediction models are series-parallel diagrams, tree diagrams, truth tables, and state-space diagrams. Although computers are not suited to producing the prediction model itself, they can assist in performing certain of the analyses required to determine what the reliability logic diagram should be; for example, the ECAP program described earlier can be used in FMEA.

The second direction in which the model-building can proceed is to establish the logic required to analyze the total system throughout the total operational profile where the system configuration or the environment (or both) can be changing. The logic must then relate the reliability events that occur in sequence, where each event may represent some characteristic of overall functional operation of a different system configuration. This type of representation leads to a combined functional and logic mode; this type of model will be called an event sequence prediction model. The development of this combined diagram generally is done manually.

An extension of the first approach to developing a prediction model is to consider the system repairable so that different states may be introduced. This leads to the state-space diagram approach, but here again, the model-building task is primarily a manual one.

All of the above approaches to prediction modeling are described in detail in Vol. IV - Prediction of this series.

7.2 Making the Reliability Prediction

It is noted here that the prediction computations are usually of the simpler types,i.e., failures of individual elements are assumed independent, and the failure probabilities of the individual elements are combined according to the simple series and parallel logic for a fixed configuration. An individual element probability is typically expressed as a discrete probability or as a failure rate with an adjustment factor (called a K factor) based on the environment.

Having derived the reliability prediction model, its use depends on the approach taken for prediction. One much-used approach is (1) derive a Boolean algebraic expression relating the events, (2) apply the fundamental laws of probability to this expression to get a prediction equation which expresses the probability of the outcome events in terms of the probabilities of the individual events, and (3) apply the prediction equation via the computer.

Another approach used frequently is to use the prediction model as a basis for Monte Carlo simulation of the system. This requires assigning appropriate numbers to represent the probabilities of each event. For example in Fig. 7-2, the event A (which is the event that element A works) is assigned probability P(A) = 0.68 and event $\overline{A}$ (the complement of A or the event that element A does not work), probability $P(\overline{A}) = 1-P(A) = 0.32$. The computer then starts a path searching process starting with element A. A random number between 0 and 1 is obtained from a random number generator routine. The computer is programmed to sequence to element B if the number is less than 0.68 and to element D if greater than 0.68. Whichever element is called up is treated in like manner using the appropriately assigned probabilities for the events associated with the elements, and thence through the network. When a terminal event, either H or G in this case, is reached it is merely tallied as a hit. Repeated trials of this procedure, starting each time from element A, will yield scores for all possible outcomes. With enough runs, the ratio of the tally for a particular outcome to the total number of trails will provide an estimate of the probability of the particular outcome occurring. The validity of this estimate depends on the validity of the numbers representing the probability of occurrence of each event. This approach is well suited to complex systems where system events occur in sequence and may represent different system configurations.

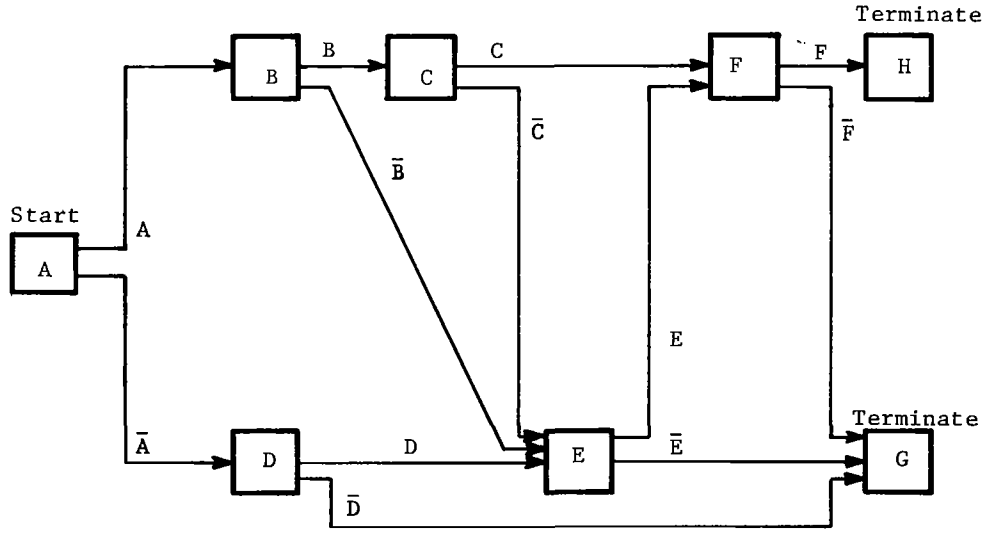


Figure 7-2. A Simple Prediction Model

As noted earlier, it is possible to include more than two states for each element used in the model. For example, in addition to considering only nominal and excess pressure in booster engines as events leading to normal launch operation and escape tower rocket ignition respectively, low engine pressure might be another state, causing engine shutdown. The only limit on complexity of the prediction model is computer size and acceptable computing time. However, logic diagrams that appear simple can be deceptive in the amount of computing time they require in performing the path searching . Simulation of complex systems is always costly, and when many outcomes are possible it may take hundreds of runs to realize each at least once. Checking out a program of this type is difficult because discrepancies can be due either to system logic or to the program. The guiding rule here is to start simple, with only several elements to represent the total system, and expand to include more logic detail as required.

7.3 Reliability Prediction Programs

Numerous computer programs for reliability prediction have been described in the literature [Refs. 7-1 to 7-14]; relatively little is known about their availability and suitability. Table 7-1 lists some of these programs. Two reliability prediction programs have been developed in connection with this report preparation. These programs with examples of their uses are discussed.

7.3.1 A Computer Program for System Reliability

One of the difficulties associated with obtaining reliability estimates for complex systems is that of evaluating precisely a prediction equation which expresses all possible events of interest. One way to alleviate this difficulty is to obtain prediction equations which provide bounds on the system reliability rather than the reliability itself. A method for doing this, on which the computer program given in Appendix B and discussed below is based, is developed in Vol. IV - Prediction of this report series. For the convenience of the reader that development is reproduced here.

In the last few years several papers have been written on the subject of reliability approximations and bounds by using the concepts of success paths (or tie sets) and cut sets. Further discussion of bounds and approximations are given by Messinger [Ref. 7-15]. A few of the more important results are given here.

The success probability of a system, typically called the system reliability, is defined as the probability of successful function of all of the elements in <u>at least one tie set</u> or the probability that <u>all cut sets</u> are good. A <u>tie set</u> or success path is a directed path from input to output as indicated in the simple system in Fig. 7-3. The tie sets or success paths are 2, 5; 1, 3, 5; and 1, 4, 5, respectively. A <u>cut set</u> is a set of elements which literally cuts all success paths or tie sets. One is normally interested in the minimal cut set; i. e., the smallest or minimal set

Table 7-1

Programs in the Reliability Prediction Area

Program Code	Program Description	Organizations (Originator or User/Sponsor)	References
CRAM	Computerized Reliability Assessment Method	ARINC/NASA	7-1
RESCRIPT	Not a specific program but a reliability-oriented programming language for prediction	Computer Concepts, Inc./?	7-2
RP-RI	Reliability Prediction of systems by combining failure rates	Radiation Inc./?	7-3
RP-LG	Reliability Prediction of systems by combining failure rates	Lockheed-Georgia/?	7-4
RP-MEL	Reliability Prediction of systems by programmed prediction equation	Marine Engineering Lab.	7-5
RP-G	Reliability Prediction and Crew Safety Analysis for complex aerospace systems from input logic models	Grumman/NASA	7-6
RP-MB	Reliability Prediction program for computing mission success and crew safety for Gemini Launch Vehicle; prediction equations required	Martin-Baltimore/?	7-7
RP-AF	Reliability Prediction by simulation	Air Force Institute of Technology	7-8
SOAR-II	Special purpose program for prediction of Apollo mission success by simulation	GE-Tempo/NASA	7-9
RAPID	Reliability Analysis and Prediction Independent of Distributions	Lear Siegler/NASA	
ARMM	Automatic Reliability Mathematical Model	NAA/?	7-10
RP-NAA	Reliability Prediction of space vehicle by Monte Carlo simulation	NAA/NASA	7-11
SFRS-W	Simulation of Failure-Responsive Systems	Westinghouse/NASA	7-12
R#14-SBC	Reliability program; computer success probability several components; different distributions; includes correlation between lifetimes	Service Bureau Corp.	7-13
R#16-SBC	Reliability program; computer system reliability estimates of components	Service Bureau Corp.	7-13
MARSEP	Mathematical Automated Reliability and Safety Evaluation Program	Mathematica/Sandia	7-14

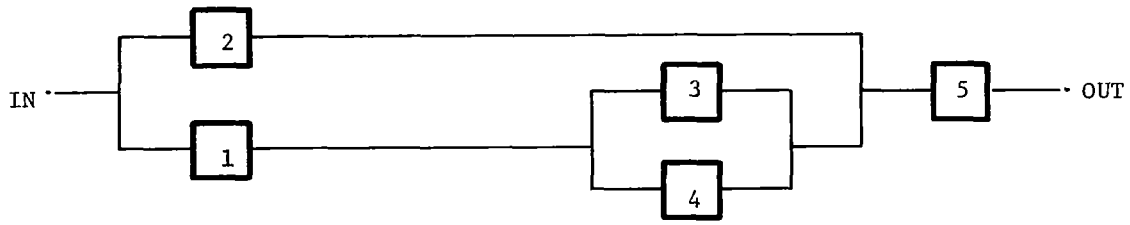


Figure 7-3A. Simple Functional Logic Diagram

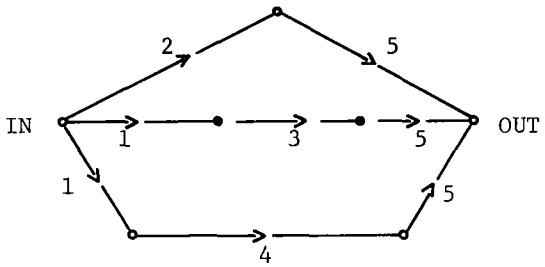


Figure 7-3B. Reliability Graph Corresponding to Functional Logic Diagram

of elements such that the elimination of any one element would no longer make it a cut. This is because a nonminimal cut set corresponds to more element failures than are required to cause system failure. In the above example the minimal cut sets are 1, 2; 2, 3, 4,; 5. Note that 1,5 is not a minimal cut set since 5 is already a cut set and is a subset of 1, 5. A cut set cuts the line of communication between input and output. A cut set is good if at least one of its elements is operative. The system failure probability or system unreliability is the probability that all tie sets are bad (a tie set is bad if at least one element fails) or the probability that at least one cut set is bad (that is, all its elements are bad). Hereafter, cut set will usually mean minimal cut set.

Let T_i, i = 1,..., I denote the tie sets, I in number; and C_j, j = 1,..., J denote the cut sets, J in number. The above statement for system reliability R can be expressed as follows.

$$R = P\{T_1 + T_2 + \ldots + T_I\} = P\{\text{at least one tie set is good}\} \qquad (7\text{-}1)$$

or

$$R = P\{C_1 \cdot C_2 \ldots C_j\} = P\{\text{all cut sets are good}\}. \qquad (7\text{-}2)$$

Equivalently the unreliability is expressed as

$$1 - R = P\{\bar{T}_1 \cdot \bar{T}_2 \ldots \bar{T}_I\} = P\{\text{all tie sets are bad}\} \tag{7-3}$$

or

$$1 - R = P\{\bar{C}_1 + \bar{C}_2 \ldots + \bar{C}_J\} = P\{\text{at least one cut set is bad}\}. \tag{7-4}$$

The above are exact formulas for the system reliability and unreliability. Bounds can be obtained by using the basic probabilistic inequalities given below.

$$R = P\{T_1 + T_2 + \ldots + T_I\} \leq \Sigma\, P\{T_i\}, \tag{7-5}$$

$$R = P\{T_1 + T_2 + \ldots + T_I\} \leq \Sigma\, P\{T_i\} - \sum_{i_1<i_2} P\{T_{i_1} T_{i_2}\}, \text{etc.} \tag{7-6}$$

Thus an upper and a bound R_{U1} lower bound R_{L1} to the reliability are respectively

$$R_{U1} = \Sigma\, P\{T_i\} \tag{7-7}$$

$$R_{L1} = \Sigma\, P\{T_i\} - \sum_{i_1<i_2} P\{T_{i_1} T_{i_2}\}. \tag{7-8}$$

In the same manner another upper bound is obtained,

$$R_{U2} = \Sigma\, P\{T_{i_1}\} - \sum_{i_1<i_2} P\{T_{i_1} T_{i_2}\} + \sum_{i_1<i_2<i_3} P\{T_{i_1} T_{i_2} T_{i_3}\} \tag{7-9}$$

The summations are over all possible combination of the subscripts taken 2 at-a-time, 3 at-a-time, etc.

Similarly the inequalities (7-5) and (7-6) can be applied to the cut-set form of the equation for unreliability (7-4) to obtain

$$1 - R \leq \Sigma P\{\bar{C}_j\}$$

or

$$R \geq 1 - \Sigma P\{\bar{C}_j\} = R_{L2} \tag{7-10}$$

and by using two terms

$$R \leq 1 - \Sigma P\{\bar{C}_j\} + \sum_{j_1<j_2} P\{\bar{C}_{j_1} C_{j_2}\} = R_{U3} \tag{7-11}$$

Example: Consider the reliability graph given in Fig.7-3. Assume independence between items and let the probabilities of success for each of the items be $p_1 = 0.93$, $p_2 = 0.86$, $p_3 = 0.92$, $p_4 = 0.95$, $p_5 = 0.98$. The probabilities for the ties and cuts are as follows:

$$P\{T_1\} = P\{2\ 5\} = 0.8428$$

$$P\{T_2\} = P\{1\ 3\ 5\} = 0.8385$$

$$P\{T_3\} = P\{1\ 4\ 5\} = 0.8658,$$

and

$$P\{C_1\} = 1 - P\{\bar{1}\ \bar{2}\} = 1 - .0098 = 0.9902$$

$$P\{C_2\} = 1 - P\{\bar{2}\ \bar{3}\ \bar{4}\} = 1 - 0.00056 = 0.99944$$

$$P\{C_3\} = 1 - P\{\bar{5}\} = 1 - 0.02 = 0.98.$$

Upper and lower bounds for the reliability are given by using Eqs. (7-7), (7-8), (7-9), (7-10), and (7-11), respectively,

R_{U1} = $P\{T_i\} > 1$ (not useful as $R_{U1} \leq 1$.)

R_{L1} = $0.843 + 0.838 + 0.866 - P\{1\ 2\ 3\ 5\} - P\{1\ 2\ 4\ 5\} - P\{1\ 3\ 4\ 5\}$

= 0.2848

R_{U2} = $0.2848 + 0.6850 = 0.9698 = R$ (This result should be equal to the system reliability)

R_{L2} = $1 - P\{\bar{C}_j\} = 1 - 0.03036 = 0.96964$

R_{U3} = $1 - 03036 + 0.00024 = 0.96988.$

As stated by Messinger [Ref. 7-15] the bounds based on the cuts sets are best in the high reliability region and those based on the tie sets are best in the low reliability region. Hence the bounds R_{L2} and R_{U3} are the preferred bounds in the above example and R_{U2} in this case saves no computation as it is the exact probability of system success, as there are only three tie sets and the bound uses all combinations of tie sets up to and including three sets.

In more general problems in which there are J cut sets the number of terms to be obtained in the lower and upper bounds computations are J and J(J-1)/2 respectively. This is compared to 2^J-1 terms obtained by expanding either Eq. (7-1) or (7-4) using tie sets or cut sets respectively.

Program Description

The bounds for system reliability, previously discussed, are obtained from calculations which are based on cut sets. This program calculates upper and lower bounds

using the probabilities of success of each item in the system. The program is written in FORTRAN. A flow diagram is given in Fig. 7-4;a program listing is in Appendix B.

Input simplicity is one of the features of this program. The user need only supply the success probabilities and a precedence list for each item in the system. The precedence is established by feeding to the computer via cards a list of items responsible to the i-th item. Table 7-2 shows an example corresponding to the reliability logic diagram in Fig. 7-3.

The algorithm is not complex, but is rather a series of simple steps. These steps in order are: read the precedence list, develop the tie sets, develop the cut sets, and calculate the bounds.

The precedence list is converted to the success paths or tie sets by a subroutine called PATH.* The arguments are: N, number of items in the system; NP, number of success paths found; IP, the array of the success paths. The precedence list is read by the PATH subroutine; its format is discussed under the input description. After being printed the paths are converted to a Boolean array of zeros and ones, and the cut sets are developed by the procedure given below. When the cut sets are available the bounds are calculated by a procedure in Ref. 7-15.

Generation of Cut Sets

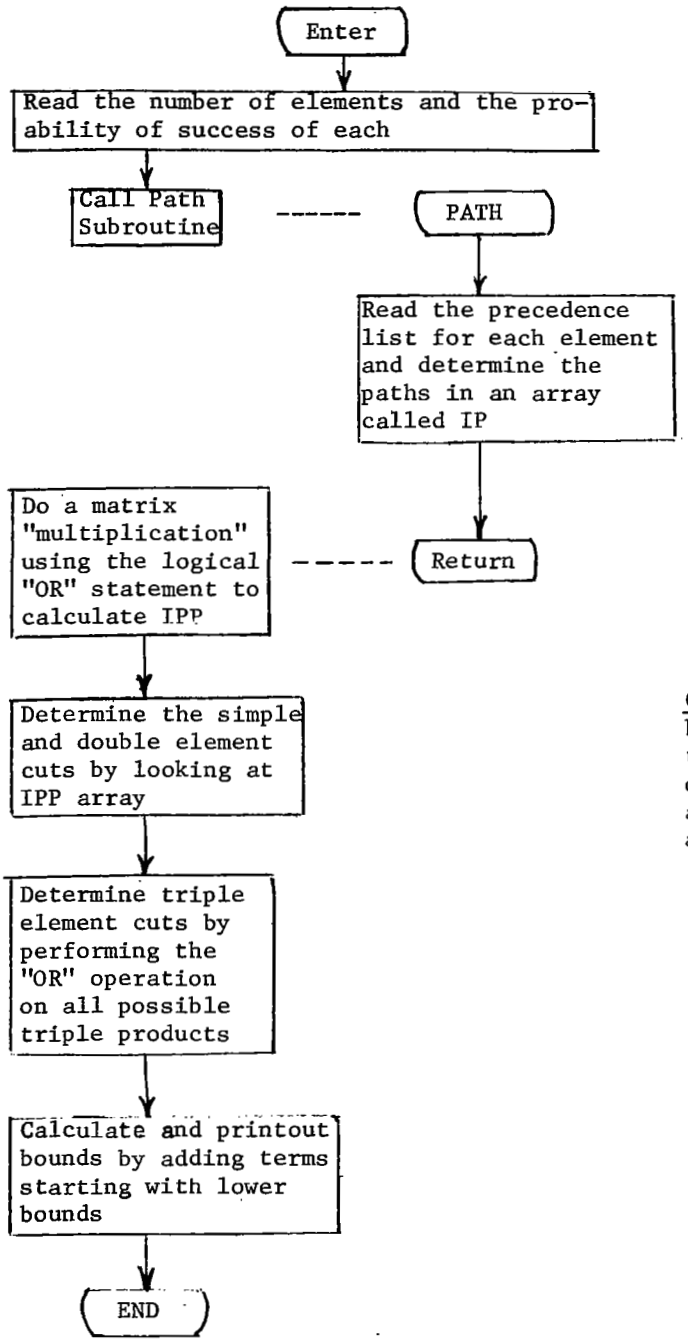
A simple procedure using Boolean logic is used for obtaining a matrix identifying the minimal cuts of the system from one containing the paths. Let the path matrix be

Table 7-2

Precedence List for Program Input

ITEM	PREDECESSORS	CARD CODE
1	IN	-1
2	IN	-1
3	1	1
4	1	1
5	2, 3, 4	2, 3, 4
OUT	5	20

*This algorithm was obtained from Naval Applied Science Labs of Brooklyn, N. Y.



Comment: The paths are determined by element number in reverse order; therefore, the program corrects the order for output purposes and forms a Boolean matrix whereby the paths are the rows.

Figure 7-4. Flow Diagram for Computer Program--Bounds for Reliability.

$$P = \begin{vmatrix} p_1 \\ p_2 \\ p_3 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{vmatrix},$$

where the paths are $p_1 = 1, 3, 5$; $p_2 = 2, 5$; and $p_3 = 1, 4, 5$, respectively. Now consider the column vectors 1, 0, 1; 0, 1, 0; etc. of the path matrix P. For a single element to be a cut, it must be in each path; i. e., its column vector in P must be the unit vector (1, 1, 1). Note that element 5 is the only element which is contained in all paths; hence 5 is the only single element cut. In general, if P_c denotes a column vector of an n-path matrix, then for

$$P_{c_i} = 1 \text{ , for all } i=1, 2,\ldots,n,$$

the corresponding element c is a single element cut. If $P_{c_i} = 0$ for some i in each path then there are no single element cuts and one must proceed to look for two element cuts.

For two element cuts consider for $c \neq d$

$$P_{c_i} + P_{d_i}$$

where the "+" indicates the logic sum or union. If

$$P_{c_i} + P_{d_i} = 1, \quad \text{for all } i=1,2,\ldots,n,$$

then elements c and d form a two element cut.

This procedure continues until all possible cuts of order 1, 2,..., n have been exhausted or until only unit vectors are obtained in the vector unions as described. At each stage all the nonminimal cuts are eliminated by using the following approach. After a possible cut of order M has been identified, it is checked against all cuts of order M-1, M-2,..., 1 by using Boolean logic for intersection, i.e., the AND operation, for the multiplication of two vectors; if the possible cut contains a cut of smaller order the vector product would be equal to the order of the smaller cut. All cuts are eliminated for which this vector product as defined is equal to the order of the smaller cut.

The above steps describe how the program identifies minimal cuts, even to the "OR" logic used to form the vector union.

Input and Program Limitations

There are three basic inputs to the program. The input variables are N, the number of items in the system, PROB, the probabilities of success of each item, IACTIV, the i-th item, and IPRED, the item(s) immediately preceding the IACTIV item.

The limit on the number, N, of items is 20 not including the end points; for N the format is (I5). There is no limit on PROB; however, each item should have specified a probability of success; format is (8E10.4).

IACTIV and IPRED are variables associated with the precedence list. IACTIV is the item actively under consideration, and IPRED is a vector of items that precede the item IACTIV. If the item IACTIV is preceded by the input point IPRED is the single number -1, and if succeeded by the output point it is the number 20. IACTIV may be any number up through 20; the IPRED vector may have at most 9 numbers. There will be N+1 input cards, one for each element in the logic model and one for the output node; the input element format is (10I5).

Output

The output is brief and easily read. Input probabilities are printed and followed by the tie sets and cut sets.

Since the calculation for bounds is done by adding terms to a series with each new term resulting in a new bound, either lower or upper, the bounds are given at each step with the appropriate last term shown. For small systems the exact system reliability is calculated before the program is terminated.

Example: The example in Fig. 7-3 is used.

The path matrix is given by

		1	2	3	4	5	Paths
		1	0	1	0	1	1,3,5
P	=	0	1	0	0	1	2,5
		1	0	0	1	1	1,4,5

and the cut matrix by

		1	2	3	4	5	Cuts
		0	0	0	0	1	5
C	=	1	1	0	0	0	1,2
		0	1	1	1	0	2,3,4

The three cuts are thus 5;1 and 2; and 2,3, and 4. The upper and lower bounds are obtained as indicated in the previous discussion. The program results as shown in Table 7-3. have been retyped from the computer printout.

Table 7-3

Bounds for System Reliability Example

CIRCUIT CONTAINS 5 ELEMENTS

ELEMENT NUMBER	PROBABILITY OF SUCCESS
1	.9300
2	.8600
3	.9200
4	.9500
5	.9800

TIE SETS OR SUCCESS PATHS (3)

PATH	ELEMENT NUMBERS
1	2 5
2	1 3 5
3	1 4 5

CUT SETS (3)

1	5
2	1 2
3	2 3 4

LOWER BOUND IS .96964E 0 LAST TERM .30361E -1

UPPER BOUND IS .96988E 0 LAST TERM .24641E -3

LOWER BOUND IS .96988E 0 LAST TERM .78407E -6

SYSTEM RELIABILITY .98988E 0

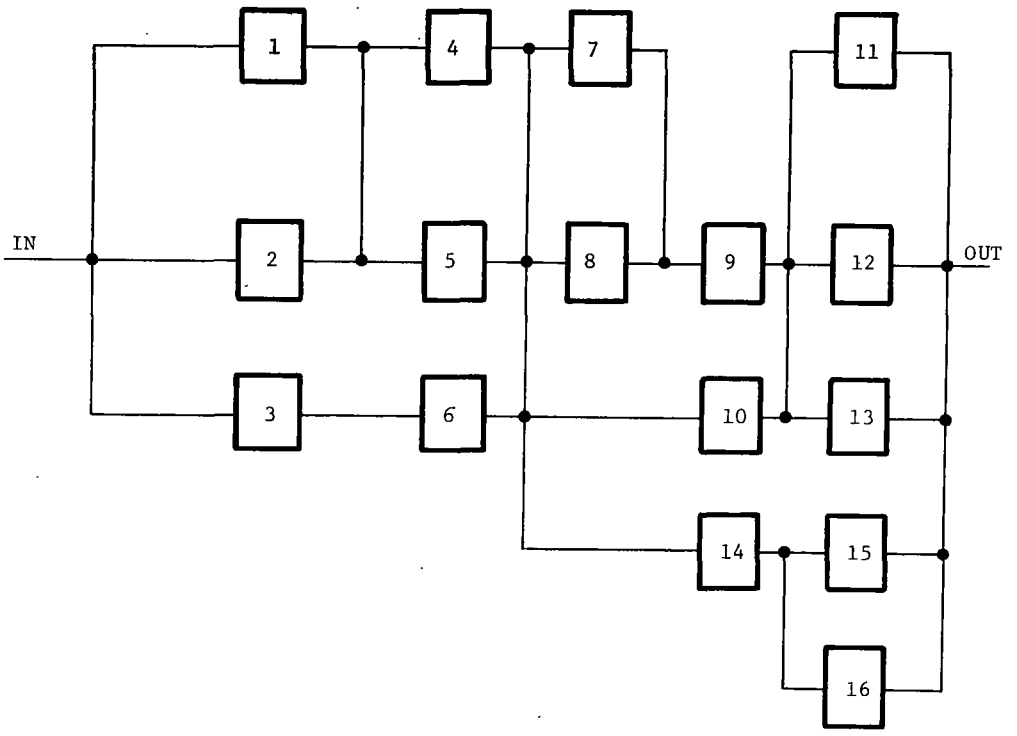


Figure 7-5. System Diagram for Bounds Program Example 2

Example 2

The system shown in Fig. 7-5 is used; it is a relatively complex series-parallel network. The input to the system is at the left of the figure. As can be seen from the figure, there are many possible success paths through the system, and hand calculation of system reliability would be at best very tedious. The reliability of each element is given in Table 7-4. As required by the program, element failures are assumed independent. The bounds program printout follows. As can be seen from the last two lines of the printout, the program has bounded the system reliability. Since the upper and lower bounds have converged to the same value, 0.97726, this value is the system reliability to 5-place accuracy.

Table 7-4

Reliabilities of Elements in Fig. 7-5

El. No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Rel.	.80	.80	.90	.85	.75	.87	.82	.82	.89	.88	.85	.85	.85	.75	.70	.70

Table 7-5

Program Printout for Example 2

BOUNDS FOR SYSTEM RELIABILITY

CIRCUIT CONTAINS 16 ELEMENTS

ELEMENT NUMBER,	PROBABILITY OF SUCCESS
1	.8000
2	.8000
3	.9000
4	.8500
5	.7500
6	.8700
7	.8200
8	.8200
9	.8900
10	.8800
11	.8500
12	.8500
13	.8500
14	.7500
15	.7000
16	.7000

TIE SETS OR SUCCESS PATHS (55)

PATH	ELEMENT NUMBERS				
1	1	4	7	9	11
2	1	4	7	9	12
3	1	4	7	9	13
4	1	4	14	15	
5	1	4	14	16	
6	1	4	10	11	
7	1	4	10	12	
8	1	4	10	13	
9	1	4	8	9	11

Table 7-5 (Cont'd)

10	1	4	8	9	12
11	1	4	8	9	13
12	1	5	14	15	
13	3	6	14	15	
14	1	5	14	16	
15	3	6	14	16	
16	1	5	10	11	
17	3	6	10	11	
18	1	5	10	12	
19	3	6	10	12	
20	1	5	10	13	
21	3	6	10	13	
22	1	5	7	9	11
23	3	6	7	9	11
24	1	5	7	9	12
25	3	6	7	9	12
26	1	5	7	9	13
27	3	6	7	9	13
28	2	4	14	15	
29	2	4	14	16	
30	2	4	10	11	
31	2	4	10	12	
32	2	4	10	13	
33	1	5	8	9	11
34	3	6	8	9	11
35	1	5	8	9	12
36	3	6	8	9	12
37	1	5	8	9	13
38	3	6	8	9	13
39	2	5	14	15	
40	2	5	14	16	
41	2	5	10	11	
42	2	5	10	12	
43	2	5	10	13	
44	2	4	7	9	11
45	2	4	7	9	12
46	2	4	7	9	13
47	2	4	8	9	11
48	2	4	8	9	12
49	2	4	8	9	13

Table 7-5 (Continued)

50	2	5	7	9	11
51	2	5	7	9	12
52	2	5	7	9	13
53	2	5	8	9	11
54	2	5	8	9	12
55	2	5	8	9	13

CUT SETS(10)

1	1	2	3		
2	1	2	6		
3	3	4	5		
4	4	5	6		
5	9	10	14		
6	7	8	10	14	
7	9	10	15	16	
8	11	12	13	14	
9	7	8	10	15	16
10	11	12	13	15	16

LOWER BOUND IS .97522E 0 LAST TERM .24782E -1

UPPER BOUND IS .97738E 0 LAST TERM .21627E -2

LOWER BOUND IS .97726E 0 LAST TERM .14357E -3

UPPER BOUND IS .97726E 0 LAST TERM .33038E -4

LOWER BOUND IS .97726E 0 LAST TERM .64429E -6

7.3.2 Reliability Cost Trade-Off Analysis Program

The REliability Cost Trade-off Analysis (RECTA) program obtains an optimum configuration for a system containing spare, active and standby components. The system configuration initially contains no redundancy involving identical elements, but may have redundant elements with different failure-rate characteristics. This program combines some of the features of those described in Refs. 7-16 and 7-17. The program is listed in Appendix C.

The main feature of the program is a subroutine which calculates the reliability of an element containing:

(1) n identical active parallel items, at least n_o of which must operate,

(2) m identical spares, and

(3) r identical standby redundant items.

The computation assumes independence and the exponential failure time distribution. Volume IV - Prediction of this series contains a complete description of the procedure. The system reliability model gives the probability of successful operation of the system in terms of the element reliabilities. The system reliability is calculated by a model supplied in a subroutine by the user. The user also supplies indicators for each element for the types of redundancy he wishes to consider; a one (1) indicates that the particular form of redundancy is permitted, and a zero (0) indicates that no items of the particular redundancy type may be added.

One additional feature of the program is the handling of majority voting logic. An upper limit is supplied as the indicator input value. The items will be incremented in steps of 1, 3, 5, ..., N where N is the limit provided by the user. An example of majority voting is for 5 items in an element, at least 3 of which must work.

Starting with the initial system configuration, all possible single item additions (two items in the case of majority voting elements) are made and the increase in the system reliability is obtained for each configuration by the element reliability subroutine and the subroutine supplied by the user for the computation of the reliability of the system. The increase in cost is also computed for each configuration using the input cost information. The ratios of the increase in reliability to the increase in cost are computed for each possible alternative as specified by the indicators. The redundant item yielding the greatest ratio is the one selected for addition and the procedure is repeated for the next step starting with the new configuration.* The program continues until a convergence criterion, supplied by

---

* This algorithm yields an incomplete undominated sequence of optimal solutions in the case of a serial system initially. In the case of nonserial systems the procedure may not yield an optimal sequence of solutions although it would be expected to yield near optimal configurations. See Ref. 7-18 concerning this point for serial systems.

the user, has been satistified. For example, it may continue until the increase in reliability is less than 0.001.

By virtue of the indicators a feature of this program which is not obvious is that it can be used for a spares allocation procedure based on either one of two criteria:

(1) minimize stockout probability subject to a given cost, or

(2) maximize system reliability subject to a given cost.

In the latter case the system configuration is used in the reliability computation whereas in the former the elements are considered to be in series.

Input Description

The input is straightforward with one optional input. A brief explanation is given for each input card and its variables; these are followed by an example.

The first two cards identify the system being analyzed with the first card having two system parameters NEL and CONVG. NEL is the number of elements in the system and CONVG is the system reliability convergence criterion. When the increase in reliability is less than CONVG the program branches to read new data. The second card has an identification for the problem being run; all 80 columns may be used and the message is not restricted as to type of characters.

The information for each element is next read in the order specified in the system model. The element parameters are defined in the following table.

Table 7-6

Input Card Variable Names

	Variable	Identification
Card 1	TIME	Length of mission
	FRATE	Failure rate
	RELSW	Switch reliability
	ELCST	Active item cost
	SPCST	Spare cost
	SWCST	Switch cost
	RSCST	Redundant standby cost
	NO	Minimum number of items necessary for operation
Card 2	IND	Indicators of type of redundancy permitted
	INPRM	Initial number of items in the system.

The option mentioned above concerns the variable IND which may indicate majority voting. If this is desired an upper limit is inserted as the indicator. The program will eliminate the particular variable from consideration when it has built up to the specified limit. The majority voting applies to active items only.

Table 7-7

Example of Input Cards

```
    9 .1000E-02
MAJORITY VOTING LOGIC WITH REDUNDANT STANDBYS IN THE LAST TWO ELEMENTS
  .1000E 03 .5130E-03 .9900E 00 .2000E 01 .2000E 01 .2000E 00 .2000E 01    1
     3    0    0    1    0    0
  .1000E 03 .5130E-03 .9900E 00 .2000E 01 .2000E 01 .2000E 00 .2000E 01    1
     3    0    0    1    0    0
  .1000E 03 .5130E-03 .9900E 00 .2000E 01 .2000E 01 .2000E 00 .2000E 01    1
     3    0    0    1    0    0
  .1000E 03 .5130E-03 .9900E 00 .2000E 01 .2000E 01 .2000E 00 .2000E 01    1
     3    0    0    1    0    0
  .1000E 03 .1054E-02           .1000E 02                                  1
     3    0    0    1    0    0
  .1000E 03 .1054E-02           .4000E 01                                  1
     3    0    0    1    0    0
  .1000E 03 .1054E-02           .4000E 01                                  1
     3    0    0    1    0    0
  .1000E 03 .6931E-02 .9900E 00 .1000E 03 .1000E 03 .1000E 02 .1000E 03    1
     3    0    1    1    0    0
  .1000E 03 .2877E-02 .9900E 00 .3000E 02 .3000E 02 .3000E 01 .3000E 02    1
     3    0    1    1    0    0
```

Output Description

Initial values of the parameters and other pertinent information about the system cost are printed first for identification. The initial reliability is calculated and printed for each element separately. This is followed by a summary of the element information and the system reliability and cost for a system consisting of no redundancy.

The iteration begins by printing the element reliability with one item added where designated by indicators. One of these additions (spare, standby, or active parallel) is selected for the optimum configuration with respect to reliability and cost. The reliability of this same element is calculated with one additional item of redundancy of each type permitted. The ratios of increase in reliability to increase

of cost are compared for this element and others calculated earlier for the optimum configuration at this stage. When the optimum is found the information for this step is printed and the program proceeds to the next step. The program repeats the above procedure adding one redundant item at a time to a selected element until the convergence requirement is met.

The 3 major steps in RECTA are summarized below.

(1) Initial step: data is read and the reliability is calculated for each element at its initial state. The initial system reliability and cost are also calculated.

(2) Intermediate step: each item of each element that is allowed to vary is incremented separately and the increases in reliability and cost of the system are calculated.

(3) Iteration loop: the loop begins by choosing the configuration generated in the intermediate step that yields the best cost-reliability trade-off. The item that is added to the system is then replaced in the intermediate state by its next increment; thus, the intermediate state always is one step ahead of the system configuration.

The program continues to query the intermediate values and add components until the system reliability satisfies the convergence criterion.

Example

This example is a simplified block diagram of a computer containing nine (9) blocks (elements) assumed to be in series logic, as shown in Fig. 7-6. All elements

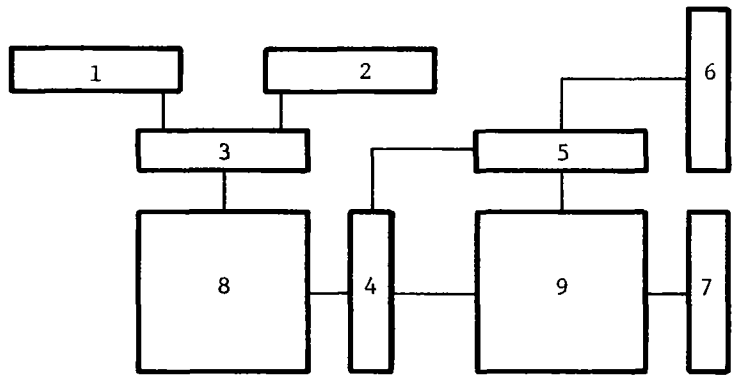


Figure 7-6. Simplified Computer Block Diagram for RECTA Example

Table 7-8

RECTA Program Example

RELIABILITY COST TRADE-OFF ANALYSIS (RECTA)

MAJORITY VOTING LOGIC WITH REDUNDANT STANDBYS IN THE LAST TWO ELEMENTS
NUMBER OF ELEMENTS 9

INPUT INFORMATION

ELEMENT	TIME	FAILURE RATE	SWITCH RELIAB.	ITEMS COST ACTIVE	SPARE	STANDBY	SWITCH
1	100.	.00051	.99000	2.00	2.00	2.00	.20
2	100.	.00051	.99000	2.00	2.00	2.00	.20
3	100.	.00051	.99000	2.00	2.00	2.00	.20
4	100.	.00051	.99000	2.00	2.00	2.00	.20
5	100.	.00105	0.00000	10.00	0.00	0.00	0.00
6	100.	.00105	0.00000	4.00	0.00	0.00	0.00
7	100.	.00105	0.00000	4.00	0.00	0.00	0.00
8	100.	.00693	.99000	100.00	100.00	100.00	10.00
9	100.	.00288	.99000	30.00	30.00	30.00	3.00

INITIAL SYSTEM CONFIGURATION

ELEMENT	INITIAL ITEMS ACTIVE	SPARES	STANDBY	INDICATORS ACTIVE	SPARES	STANDBY
1	1	0	0	3	0	0
2	1	0	0	3	0	0
3	1	0	0	3	0	0
4	1	0	0	3	0	0
5	1	0	0	3	0	0
6	1	0	0	3	0	0
7	1	0	0	3	0	0
8	1	0	0	3	0	1
9	1	0	0	3	0	1

Table 7-8 (Cont'd)

RELIABILITY ESTIMATES FOR ELEMENT 1 CONTAINING

IDENTICAL ITEMS IN PARALLEL	1
MINIMUM NUMBER OF ITEMS FOR OPERATION	1
IDENTICAL SPARES	0
IDENTICAL ITEMS IN STANDBY REDUNDANCY	0
SWITCH RELIABILITY	.9900
FAILURE RATE	.0005
TIME	100.0

RELIABILITY FOR FIXED INACTIVE REDUNDANCY

STANDBY	REL.
0	.949993E 0

ELEMENT RELIABILITY .949993E 0

INITIAL STEP

SYSTEM CONFIGURATION

ELEMENT	* ITEMS * ACTIVE	SPARE	STANDBY	RELIABILITY
1	1	0	0	.949993
2	1	0	0	.949993
3	1	0	0	.949993
4	1	0	0	.949993
5	1	0	0	.899963
6	1	0	0	.899963
7	1	0	0	.899963
8	1	0	0	.500023
9	1	0	0	.749987

SYSTEM RELIABILITY .222636
SYSTEM COST 156.00

Table 7-8 (Cont'd)

RELIABILITY ESTIMATES FOR ELEMENT 9 CONTAINING

IDENTICAL ITEMS IN PARALLEL	3
MINIMUM NUMBER OF ITEMS FOR OPERATION	2
IDENTICAL SPARES	0
IDENTICAL ITEMS IN STANDBY REDUNDANCY	0
SWITCH RELIABILITY	.9900
FAILURE RATE	.0029
TIME	100.0

RELIABILITY FOR FIXED INACTIVE REDUNDANCY

STANDBY	REL.
0	.843731E 0

ELEMENT RELIABILITY .843731E 0

RELIABILITY ESTIMATES FOR ELEMENT 9 CONTAINING

IDENTICAL ITEMS IN PARALLEL	1
MINIMUM NUMBER OF ITEMS FOR OPERATION	1
IDENTICAL SPARES	0
IDENTICAL ITEMS IN STANDBY REDUNDANCY	1
SWITCH RELIABILITY	.9900
FAILURE RATE	.0029
TIME	100.0

RELIABILITY FOR FIXED INACTIVE REDUNDANCY

STANDBY	REL.
0	.749987E 0
1	.965755E 0

ELEMENT RELIABILITY .963593E 0

Table 7-8 (Cont'd)

STEP 1

SYSTEM CONFIGURATION

ELEMENT	ITEMS * ACTIVE	SPARE	STANDBY	RELIABILITY
1	3	0	0	.992744
2	1	0	0	
3	1	0	0	
4	1	0	0	
5	1	0	0	
6	1	0	0	
7	1	0	0	
8	1	0	0	
9	1	0	0	

SYSTEM RELIABILITY .232655
SYSTEM COST 160.00

STEP 14

SYSTEM CONFIGURATION

ELEMENT	ITEMS * ACTIVE	SPARE	STANDBY	RELIABILITY
1	3	0	0	
2	3	0	0	
3	3	0	0	
4	3	0	0	
5	3	0	0	
6	3	0	0	
7	3	0	0	
8	1	0	4	.999018
9	1	0	3	

SYSTEM RELIABILITY .890726
SYSTEM COST 747.00

can be made redundant by using majority voting logic and two elements can be further modified by using standby elements with switching. The input information is contained on the following printout. The initial system configuration indicates the number of active, spare, and standby elements in the system at the beginning of the computer run, the indicator tells the computer the elements which can be made redundant by adding further active items or using majority voting logic, adding spares and standbys. In this example the majority voting logic is used (a maximum of three elements at least two of which must operate), no spares are permitted, and standby items are permitted for elements 8 and 9. The program output is given in Table 7-8.

The program obtains the reliability estimates for each element subject to its initial configuration; that for element 1 is shown below. Then the reliability of the initial system is computed from the model supplied by the user. The initial step is given in the printout. At this point the program is ready to alter each element in all possible manners as specified by the indicators in order to determine the optimum configuration for one item added (two for the majority voting alternative). The results for item 9 are given because there are two alternatives. The item which gives the largest increase in reliability per unit cost is the one selected for step 1; in this case it is element 1 and a majority voting element with three items is used. This procedure is repeated at each step to obtain a system configuration with the desired relaibility or one for which the increase in reliability is less than 0.001. Fourteen steps were used in the analysis; the final system configuration is given on the final printout along with the system reliability and cost.

References

7-1. Van Tijn, D. E.: Description of the Computerized Reliability Analysis Method (CRAM). ARINC Research Monograph 11, 1964.

7-2. Whiteman, I. R.: RESCRIPT--A Computer Programming Language for Reliability, Presented at Fifth Annual West Coast Reliability Symposium, Los Angeles, California, 1964.

7-3. House, J. F.; and LaCapra, John: Systems Reliability Analysis and Prediction through the Application of a Digital Computer. Presented at National Symposium on Space Electronics and Telemetry, Miami Beach, Florida, 1962.

7-4. Shelley, B. F.; and Hamilton, D. C.: A Mechanized Aircraft Reliability Analysis Model. Proceedings of the Tenth Symposium on Reliability and Quality Control, Washington, D. C., 1964.

7-5. McFaul, Charlotte: Deep Submergence Rescue Vessel Reliability Prediction. Technical Memo 415/65. US Navy MEL, Annapolis, Maryland, 1965.

References (Continued)

7-6. Weisberg, S. A.; and Schmidt, J. H.: Computer Techniques for Estimating System Reliability. Proceedings of the 1966 Annual Symposium on Reliability, San Francisco, California, 1966, pp. 87-97.

7-7. Kiefer, F. P.; et. al.: Man-rating the Gemini Launch Vehicle (Crew Hazard and Mission Analysis). Proceedings of the 1966 Annual Symposium on Reliability, San Francisco, California, 1966, pp. 87-97.

7-8. Finch, R. E.: An SPS Subroutine as a Simulation Aid. Master of Science Thesis. School of Engineering, Air University, Wright-Patterson AFB, AD 425 237, 1963.

7-9. Ottlinger, J. A.; et. al.: Survey of Studies and Computer Programming Efforts for Reliability, Maintainability and System Effectiveness. Report OEM-1, Office of the Director of Defense Research and Engineering, Department of Defense, AD 622 676.

7-10. McKnight, C. W.; et. al.: An Automatic Reliability Mathematical Model. Proceedings of the Eleventh National Symposium on Reliability and Quality Control, Miami Beach, Florida, 1965.

7-11. Hershkowitz, B. H.; et. al.: Reliability Simulation Model. Proceedings of the Tenth National Symposium on Reliability and Quality Control, Washington, D. C., 1964.

7-12. Hannigan, J. M.: A Computer Program for the Simulation of Failure-Responsive Systems. Technical Report No. 6, Westinghouse Defense and Space Center, STAR N66-26880, 1966.

7-13. House, J. F.; and LaCapra, John: Reliability Engineering at SBC. Brochure from the Service Bureau Corporation/Computing Sciences Division, 1966.

7-14. Hubbard, R. L.: The Marsep Program. Two page description from Mathematica, Princeton, New Jersey.

7-15. Messinger, M.; and Shooman, M. L.: Reliability Approximations for Complex Structures. 1967 Annual Symposium on Reliability, Washington, D. C., January 10-12, pp. 292-301.

7-16. Trott, E. P.: Maintainability and Reliability Cost Effectiveness Program (MARCEP). Fourth Annual Reliability and Maintainability Conference, July 1965, pp. 219-228.

7-17. Parr, V. B.: Automated Reliability Trade-Off Program--ARTOP II. Proceedings 1967 Annual Symposium on Reliability, January 10-12, 1967, pp. 847-850.

7-18. Barlow, R. E.; and Proschan, F.: Mathematical Theory of Reliability. John Wiley & Sons, Inc., New York, 1965, 256p.

7-19. Sandler, G. H.: System Reliability Engineering. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1963, 221p.

7-20. Meyers, R. H.: Reliability Engineering for Electronic Systems. John Wiley & Sons, Inc., New York, 1964, 360p.

8. Testing

Many of the results of experimental programs can be analyzed by graphical techniques such as drawing a curve by freehand through a set of data points, or by comparing a test measurement with a physical requirement. These particular methods of analyses do not require formal computation by the use of a digital computer. However,it is not unusual in typical experimental programs to encounter situations in which one is measuring several performance attributes and as many as 10 or more independent variables such as part characteristics and environmental stresses. In order to analyze data of this complexity it is usually necessary to use digital computer programs which are already available.

In addition, one is often faced with the problem of estimating the parameters of life distributions on the basis of an observed sample of items placed on test for a fixed test time. In order to have the capability of describing these data by means of one or more of the many failure-time distributions, it is convenient to have computer programs to perform the tedious analyses.

In this section the computational approaches are subdivided into those which pertain to: (1) attribute data, (2) variables data, and (3) stress-strength measurements. By attribute data we mean simply that the observation of an experiment is classified as a failure or nonfailure, or in a case of a performance measurement that the observation is classified as go or no-go. In the latter case, the region of observations is subdivided into two disjoint regions; the acceptable performance region and the nonacceptable region. By variables data we mean observations which can take on any one of a set of values over a given range of values. The third category, stress-strength measurements includes stress-at-failure data, such as would be obtained in a tensile test of a particular metal specimen. It also includes the data resulting from sensitivity testing, where an item is placed on test at a fixed stress level and test results recorded as a failure or a nonfailure.Table 8-1 summarizes the results of this section with respect to the type of data and the associated problems. Table 8-2 contains a listing of the computer programs which may be helpful in solving the corresponding problems.

8.1 Attribute Data

The typical computational problems associated with attribute data are to provide sampling plans and their operating characteristics and to obtain confidence limits for the true proportion of nonfailures (or "go" items). Both of these problems usually are solved using the binomial distribution.* It is necessary to sum several terms of this distribution in order to obtain the probability that a given sampling plan

*A description of distributions is given in the Appendix of Vol. III - Testing of this series.

Table 8-1

Categories of Testing Data and Associated Computational Problems

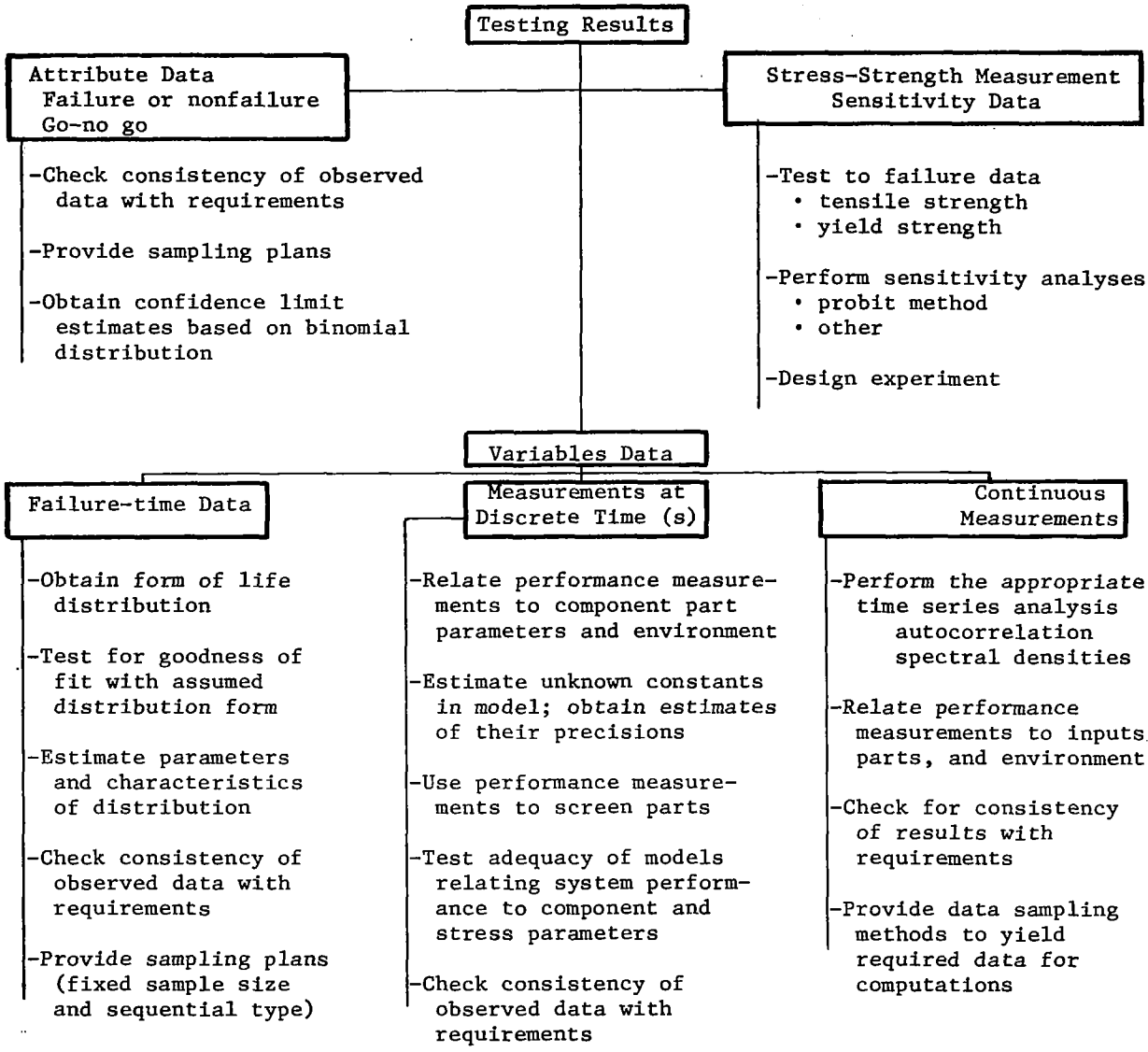


Table 8-2

Testing-Related Computer Programs with Corresponding Problem Areas

1. Attribute Data

   A. Library of Programs

      (1) BMD (Ref. 8-16)
      (2) STAT-PACK (Ref. 8-7)

2. Variables Data

   A. Performance Data (at Discrete Times)

      (1) Least Squares (Linear)

         (a) STAT-PACK (Ref. 8-7)
         (b) WHIRLPOOL (Ref. 8-2)
         (c) IBM, 6.0.057 (Ref. 8-3)
         (d) BMD programs (Ref. 8-16)

      (2) Nonlinear Least Squares

         (a) NOLLES (Ref. 8-1)
         (b) SDA-3094 IBM-SHARE Library

      (3) General Reference (Ref. 8-1)

   B. Performance Data (Continuous Records) - Autocovariance and Power Spectrum

      (1) STAT-PACK (Ref. 8-7)
      (2) BMD programs (Ref. 8-16)

   C. Failure-Time Data

      (1) Distribution Free Estimates

         Burn-in Process, Estimation of Hazard Data and Lifetime, Density Function (Ref. 8-15)

      (2) Estimate of Parameters of Assumed Distributions

         (a) Weibull (Ref. 8-13)
         (b) Gamma (Ref. 8-8, 8-13, 8-14)
         (c) Extreme Value (Ref. 8-11)
         (d) Log-Normal (Ref. 8-10)
         (e) Logistic (Ref. 8-12)
         (f) Normal (Ref. 8-9)

         Programs can be obtained in connection with each of the above although they may not be specifically identified in the references.

      (3) Stress Strength Measurements -- Sensitivity Data (Ref. 8-6)

will accept a lot of items given the true proportion defective in the lot. Two quality levels are chosen, one which is considered acceptable and the other considered nonacceptable. These are referred to as the acceptable quality level (AQL) and the lot tolerance percent defective (LTPD) respectively.

The probability of rejecting a lot of items given that the quality level is equal to the AQL is called the producer's risk. The probability that the lot is accepted given that the proportion defective is equal to the LTPD is called the consumer's risk. If it is necessary to compute these two risks for a number of problems, it is desirable to have a computer program to perform the necessary computations.

Many programs have been written for these problems, and the results have been tablulated in a large number of tables of sampling plans and by means of graphs [Ref. 8-1]. Listings of these programs have not been provided in the literature, primarily because such programs are easy to write.

Computations similar to those described above are necessary to obtain confidence limits for the true proportion of failures p (or defectives) in a lot of submitted items. It is often desired to obtain an upper limt p_u and to do so requires the solution of an equation of the form:

$$\sum_{x=0}^{x_o} \binom{n}{x_o} p_u^x (1-p_u)^{n-x} = \alpha \ , \qquad (8\text{-}1)$$

where

x_o is the observed number of failures,

p_u is the upper confidence limit,

n is the number of items in the sample,

$1-\alpha$ is the confidence level, and

α is the risk of not including the true proportion of defectives in the confidence interval $0 < p < p_u$ for p the true proportion of failures.

This equation can be solved by an iteration procedure using the incomplete Beta function of one of the transformed distributions such as the variance ratio of an F distribution. A program can easily be written to perform the required computation if one supplies as inputs the necessary values of the F distribution or if one provides an approximating function to the F distribution for each possible combination of its two parameters. The latter procedure would require considerable input so a simpler procedure would be to solve the equation by a direct iteration procedure.

8.2 Variables Data

It is convenient in this section to divide the variables data into the three categories:

(1) failure-time data,

(2) performance measurements at discrete time(s), and

(3) continuous recording of performance measurements.

This classification of variables data is made primarily for the convenience of the computational procedures; computer programs associated with the analyses do not necessarily match the classification of tests described in Vol. III - Testing of this series. For example, the breakdown of performance measurements into the two categories, discrete versus continuous, corresponds to the digital versus analog recording mechanisms. Although both these types of measurements are used for the same general purpose, the analytical methods are quite different.

8.2.1 Failure-time Data

If a sample of items are placed on test for a fixed test time or until a specified number of failures has occurred, the test results consist of a set of failure times for the failed items and the terminated test time for all items which have not failed. It is usually desired to predict, on the basis of these data, the behavior of a large collection of items to be used under similar conditions. When performing this prediction, certain problems must be considered:

(1) discriminate between the forms of the life distributions, e.g., normal, exponential, Weibull, etc.,

(2) test for goodness of fit with an assumed distribution form,

(3) estimate the parameters of the distribution, e.g., the failure rate parameter in the case of the exponential distribution,

(4) estimate characteristics of the failure rate distribution,

(5) provide testing plans and their associated operating characteristics, and

(6) check consistency of observed data with contractual requirements.

Some techniques for discriminating between the forms of the life distributions have been given in the literature, for example, see Refs. 8-4 and 8-5. However,it is possible to compute criteria for goodness-of-fit for each of the distributions and select the particular form giving the best value of this measure. Some statistical programs are available for performing a goodness-of-fit, namely:

(1) Kolomogorov-Smirnov tests, and

(2) χ^2 tests.

Computer programs for these tests are included in STAT-PACK [Ref. 8-7]. This package of programs appears to be the most comprehensive package available to date.

The programs are written for small to medium size computers (8K words) and they do not require any nonstandard features. They are card input and card and/or printer output oriented. The programs are written entirely in the FORTRAN II language. The output of each program is labeled as completely as possible for ease of understanding by users.

One of the basic problems in comparing distributions is estimating the parameters of each proposed distribution. Several programs are available for estimating the parameters of the normal, log-normal, Weibull, gamma, generalized gamma, exponential, extreme value, and logistic distribution. In particular, an entire series of FORTRAN computer programs for this purpose are available upon request from the Aerospace Research Laboratory (ARL), Wright-Patterson Air Force Base, Ohio. In addition to these programs there is a collection of references describing the parameter estimation procedures for each of the above distributions. Almost all of these are available in the published literature [Refs. 8-9 through 8-14]. The estimation procedures are iterative and based on the maximum likelihood method of estimation. Four programs are included in STAT-PACK for estimating the parameters of the normal, log-normal, and the generalized gamma distributions. Some of the above programs include procedures for estimating the precision of the estimated parameters.

If one is unable to assume a particular form of the distribution, it may be possible to make an assumption concerning the monotonic behavior of the hazard rate. For example, this rate may decrease with time for many electronic components. In such cases it is desirable to estimate the hazard rate at the end of the test. A paper appeared recently [Ref.8-15] on this subject and included the listing of a program for obtaining confidence limits for the estimated failure rate at the termination of the test under the assumption of decreasing failure rate.

A great many sampling plans have been provided in the literature under the assumption that the failure time distribution takes on one of the many forms given above. The program is not normally listed in connection with the computations of the sampling plans; however, it is possible to write these programs in most cases by studying the discussions accompanying the tabulated results.

8.2.2 Performance Measurements at Discrete Time(s)

In this section performance measures such as the output voltage or the current gain of an electronic circuit or the "hot spot" temperature in a nuclear reactor core will be considered. It is assumed that one wishes to relate these performance measurements to characteristics of the component parts and the environmental stresses. Very often it is possible to write these relationships on the basis of technical knowledge concerning the circuit. On the other hand, it is sometimes possible only

to relate the performance to certain part characteristics and environmental stresses by means of an analytical expression in which certain constants or parameters are unknown but which can be estimated from the results of an experiment. Some of the problems which are typical are:

(1) to estimate unknown constants in the analytical models and obtain estimates of the precisions of the constants and of the complete model,

(2) to use analytical models to screen out the "bad" components,

(3) to check the consistency of the observed data with the contractual requirements, and

(4) to select the parts and their associated characteristics to optimize the performance of a circuit.

To estimate the unknown constants in the analytical models, one can make use of any one of many computer programs based on the method of least squares. If a model is linear in the unknown constants to be estimated there are three basic approaches which have been programmed:

(1) fitting the complete model,

(2) fitting the model by adding on terms one at a time, called step-wise regression, and

(3) fitting all combinations of linear models taking the variables one at a time, two at a time, etc.

Several programs are included in STAT-PACK for the approaches (1) and (2) given above. Two programs are available for the third approach [Ref. 8-2, 8-3]. In case the model is nonlinear in the constants to be estimated the least squares procedure is still applicable, but the method of solution is iterative and based on one of many possible searching techniques. Several programs have been written for nonlinear regression problems [Refs. 8-1, 8-7]. In addition to the above mentioned programs one will find comparable programs in the SHARE, CO-OP, and other such computer service systems. In order to estimate the precisions of the constants certain additional computations must be performed, such as obtaining the sum of squares of deviations of the observations from the predicted mean performance values, and inverting matrices. Most of the programs described above include some of these additional computational features.

One technique used to screen bad components is to obtain a linear discriminating function with the characteristics of the components. The coefficients in the linear function are estimated by an approach similar to that used in least square problems. A computer program in STAT-PACK [Ref. 8-7] is available for performing this analysis. Having determined the functional relationship an item is declared good only if, for

example, the value of the function is less than or equal to a particular constant c and is declared bad if the value of the function exceeds c. The value of the function is determined by substituting the characteristics of the components into the discriminating linear relationship with known constants estimated from the data.

In order to select the component part of a system such that the performance will be optimized, it is necessary to obtain an analytical model relating performance measures to the pertinent part characteristic and environmental stresses. Obtaining this model has been previously discussed; it is assumed that such a model has been obtained from theoretical methods and/or experimental results. Given the model, the problem then is to find the maximum or minimum value of the function for the region of possible values of the part characteristics. The many optimization programs that are available for solving these problems were tabulated and discussed briefly in the section on optimization techniques. Those techniques which would be of particular value here are the search techniques and nonlinear programming methods, because it is expected that most of the relationships will be nonlinear. The optimization techniques will yield the optimum values of the part characteristics from which one can hopefully select the best parts to use in the system.

8.2.3 Continuous Recording of Performance Measurements

In order to assess the performance of many physical systems it is often necessary to record measurements continuously by means of analog equipment. Although the use of an purpose for taking such measurements does not differ from those taken at discrete times, the analysis techniques are quite different. Hence this type of measurement is treated separately. Typical computation problems that arise in this connection are:

(1) performing time series analysis, including autocorrelation and spectral density analyses;

(2) relating characteristics of performance measurements to input, parts, environmental characteristics; and

(3) providing data record sampling methods to yield the desired results and the required degree of precision.

The usual procedure in analyzing continuous records is to select an appropriate set of data at equal time intervals from the data tape of interest. These data sets make up a time series which then become input to a standard computer program which performs the autocorrelation and spectral density analysis. Many programs are available to perform these computations; for example, STAT-PACK includes a time series analysis and a time series plotter program. The BMD package of statistical programs [Ref. 8-16] contains two applicable programs; one performs a cross-spectral

analysis and the other performs related computations. Similar programs are available through computer service organizations.

8.3 Stress-Strength Measurements

A great many testing problems fall into the category of determining the strength of the components to be used in a system. Although strength may be considered to be a performance measurement in the general sense, it is treated here in a separate section because of the nature of the testing problem and the resulting data.

It is not always possible to place an item on test and increase the strength in a continuous manner until the item fails and use the stress at the time of failure as the strength of the item. In testing many components the procedure is to place several items on test at each of several stress levels and observe the number of failures at each stress level. From these test results one can derive a distribution function for the probability of failure versus the stress level. Such testing is frequently referred to in the literature as sensitivity testing. A large variety of sensitivity tests have been discussed in the literature; these have been conveniently summarized in [Ref. 8-6]. One of the earliest sensitivity tests is a sequential procedure referred to as the Bruceton or the "up and down" test method. In this type of experiment the iterms are tested one at a time at a stress level; each item tested is dependent on the response and the stress level of the previous item tested. Many variations of these tests have been suggested, most of which are just different procedures by which one determines the stress level for each item tested in terms of the levels used for all previous tests rather than just the last-tested item.

The analyses of the data resulting from such experiments are usually quite easily performed by manual methods. Consequently, only a few programs are available for performing the analyses of test data resulting from sensitivity experiments. In particular, a program for a probit* analysis is included in the BMD series of programs [Ref. 8-16] and one in [Ref. 8-6]. The latter reference includes in addition computer programs for Monte Carlo simulation of the test results and the analysis of proportions of failures by the method of reversals. This latter method is frequently used in the analysis of experiments in which the stress level is determined on the basis of the proportion of successes observed at all previous stress levels tested, and the proportion of failures is assumed to be either an increasing or a decreasing function of the stress level.

---

*The probit method is a nonsequential design for relating response to stress or stimulus level.

References

8-1. Nelson, A. C.; et. al.: Evaluation of Computer Programs for System Performance Effectiveness. Progress Report No. 1 (Lab Project 920-72-1, SF-013-14-03, Task 1604, Contract N00140 66C 0499), Research Triangle Institute, System Statistics Group.

8-2. Krumbein, W. C.; et. al.: Whirlpool, A Computer Program for "Sorting Out" Independent Variables by Sequential Multiple Linear Regression. Northwestern University, Evanston, Illinois, 1964, AD 611 142.

8-3. IBM, 6.0.057: Linear Regression Analysis of All Combinations of Variables.

8-4. Cox, D. R.: Further Results on Tests of Separate Families of Hypotheses. University of London, J. Roy. Stat. Soc, (B), no. 24, 1962, pp. 406-424.

8-5. Cox, D. R.: Tests of Separate Families of Hypotheses. Proceedings of the 4th Berkeley Symp., vol. 1, pp. 105-123.

8-6. Rothman, D.; Alexander, M. J.; and Zimmerman, J. M. : The Design and Analysis of Sensitivity Experiments. NASA CR-62026, vols. I and II, May 1965.

8-7. Shannon, Stan; and Henschke, Claudia; STAT-PACK: A Biostatistical Programming Package. Wadley Research Institute, Dallas, Texas, Commun. ACM, vol. 10, no. 2, Feb. 1967, pp. 123-125.

8-8. Harter, H. Leon: Maximum-Likelihood Estimation of the Parameters of a Four-Parameter Generalized Gamma Population From Complete and Censored Samples. Applied Mathematics Research Laboratory, ARL 67-0089, April 1967.

8-9. Harter, H. Leon; and Moore, Albert H.: Iterative Maximum-Likelihood Estimation of the Parameters of Normal Populations from Singly and Doubly Censored Samples. Biometrika, vol. 53, nos. 1 and 2, 1966, pp. 205-213.

8-10. Harter, H. Leon; and Moore, Albert H.: Local-Maximum-Likelihood Estimation of the Parameters of Three-Parameter Log-normal Populations from Complete and Censored Samples. J. Am. Stat. Assoc., vol. 61, September 1966, pp. 842-851.

8-11. Harter, H. Leon; and Moore, Albert H.: Maximum-Likelihood Estimation, from Doubly Censored Samples, of the Parameters of the First Asymptotic Distribution of Extreme Values. Aerospace Research Labs and Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

8-12. Harter, H. Leon; and Moore, Albert H.: Maximum-Likelihood Estimation, from Censored Samples, of the Parameters of a Logistic Distribution. Aerospace Research Labs and Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

8-13. Harter, H. Leon; and Moore, Albert H.: Maximum-Likelihood Estimation of the Parameters of Gamma and Weibull Populations From Complete and From Censored Samples. Technometrics, vol. 7, no. 4, November 1965.

8-14. Harter, H. Leon: Series Expansions for the Incomplete Gamma Function and Its Derivatives. Blanch Anniversary Volume, Aerospace Research Labs, Office of Aerospace Research, U. S. Air Force, February 1967.

References (Continued)

8-15. Barlow, R. E.; Proschan, Frank;Scheuer, Ernest M., with an Appendix by Madansky, Albert: Statistical Estimation Procedures for the "Burn-In" Process. RM-5109-NASA, RAND Corporation, September 1966.

8-16. Dixon, W. J., ed.: Biomedical Computer Programs (BMD). Health Sciences Computing Facility, Dept. of Preventive Medicine and Public Health, School of Medicine, Univ. of Calif., Los Angeles, Calif., Jan. 1, 1964.

9. *Trends in Digital Computation*

In previous sections of the report, we have identified and discussed the various aspects of design for reliability where the computer can provide assistance. In this section we summarize some recent developments in communicating with the computer which promises to make it of much greater value to the scientist and the engineer. These developments are not specifically related to reliability, but since they are of a general nature their impact will certainly be felt in many future uses of the computer for reliability analyses.

There are three computer developments we wish to discuss. First, the use of problem-oriented languages is certain to spread as they continue to be developed and their utility becomes both greater and more widely appreciated. The other two developments are on-line computation and computer graphic input/output capabilities, here simply called graphics.

Problem-oriented languages are already in wide use [Refs. 9-1 to 9-4]. A problem-oriented language permits the description of a broad class of problems in a given problem area via a simple vocabulary comprised of terms familiar to the engineer working in that problem area. For example, the electronic circuit analysis program ECAP input language uses for the most part the same nomenclature to describe a circuit to be analyzed that the circuit analysis engineer would use to analyze the circuit by hand.

A computer program written in a problem-oriented language is not a program in the ordinary sense. This is because it is really just an unambiguous problem description rather than the logical sequence of steps required for the solution of the problem. The sequence (i.e., the algorithm) required to implement the solution of a given problem is incorporated as a part of the computer program for processing input statements to the problem-oriented language; these input statements are the problem description. Thus the programmer or the designer does not need to worry about whether his algorithm for solving the problem is correct; he need only worry that his problem is properly and unambiguously stated. Problem-oriented languages have already been developed for use in designing chemical processing plants [Ref. 9-1], structures [Ref. 9-2], and electrical circuits [Refs. 9-1, 9-3, and 9-4].

The only responsibility of the problem-oriented language user is that he know the syntax of the problem-oriented language and some simple rules concerning the ordering of the statements which describe the problem he is solving. To summarize, the problem-oriented language is simply a special program which allows as input the unambiguous description of a particular problem suited to that language and the associated data required for solution of the given problem. The individual using the language writes a new input program for each different problem, without having to worry about the

problem solution algorithm in contrast to procedure-oriented and assembly language programs. To provide the advantages of the problem-oriented language there must be a computer program which processes the input statements, digests the information contained in the statements, and generates the proper machine language program which when executed solves the problem described by the input statements.

On-line computation refers to the situation wherein the computer user sits at the computer console (it may be the console of a small computer just for the single programmer, or it may be a remote terminal connected to a large central computer) and views the results of his program instantaneously. In the early days of digital computation, it was a practical thing for the designer or other computer users to use the computer in this fashion. However, as the machines became bigger, more powerful, and more expensive, it became no longer practical for the individual to use the computer in this individual fashion. The result was that computer monitoring programs called operating systems were developed to supervise the operation of the machine. Computer systems operating in this mode process the computer user's program in a sequential fashion, so that each program is completely finished before the next program is begun. The mode is called batch processing. The computer facility is operated in such cases (and this is the usual case today) on a closed-shop basis, which means that the computer user is not present when his program is being run and the time delay between delivery of the program to be run and the return of the computer results (the so-called turn-around time) varies from hours to days.

Because of the turn-around problem, it simply is not practical to use the closed-shop computer to solve problems by heuristic methods, extrapolating earlier successes to obtain new ones. Because of the nature of engineering design, many of the most challenging engineering problems are most effectively solved by such methods. If the computer is to be of maximum assistance in this design process, turn-around times of hours are obviously hopelessly long. Even turn-around times of minutes are usually too long to allow the designer to use the heuristic method of solving problems while interacting with the computer.

The provision of a method for allowing the designer to interact directly with the computer can be obtained either by the small individual computer or by a remote console linked to a large computer. Although both approaches have merit, the remote terminal linked to the large computer is perhaps more valuable in this role.

Since the response time of the designer is quite slow compared to the computer, the instanteous response of the computer to the request of the designer can be provided economically only if the resources of the computer are shared for other purposes while the designer is thinking and modifying his programs, etc. It appears certain

that the near future will see the widespread use of computer consoles by engineering personnel for effective designer-computer interaction, in this time-shared mode of computing. Ref. 9-1 contains some examples of such uses.

The development of computer graphics is both a powerful additional computer capability in itself and a complement to the above-discussed new computer developments. The development of effective graphic input-output devices for computers traditionally has lagged the development of computers, and it is only quite recently that versatile graphic input-output devices have become available at reasonable cost. The first graphical output device was doubtless the line printer wherein a clever programmer used appropriately chosen characters to sketch a graph or a crude picture. Then, program-controllable cathode ray tube output devices became available. Although the early ones were quite limited in their capabilities, they provided great improvements over line printers used to produce pictures. Cathode ray tubes with graphical input capability in addition to output first became available in the early 1960's. Some such equipment allows the drawing of lines directly on the face of the scope using light pens or other input devices. Others allow only the display of characters at fixed locations only; the characters are typically input via a typewriter-like input device.

A considerable variety of improved graphical devices for computers are currently under development. These devices when possessing line drawing capabilities require quite high transfer rates between the display device and the computer to maintain picture clarity. Consequently, it is common to find a small computer whose sole job it is to maintain and manipulate the display information, connected to a large computer which performs the computations required for the problem under study.

Typical of what can be done with the combination of graphic input/output devices in a large powerful computer is the DAC system [Ref. 9-1] developed by the General Motors Research Center. This system, in addition to providing direct communication between the designer and a powerful computer, can produce control tapes for automatic drafting machines, numerically controlled milling machines, etc. Such systems appear destined to play important roles in the design of all future complex engineering systems.

The combination of all three of the above developments has already been made on an experimental basis [Refs. 9-5 and 9-6]. In these efforts circuit analysis programs were used on-line via graphic input/output devices to the computer. Those people who have used these experimental systems are highly enthusiastic about the effective increase in design capability through the use of these systems. Certainly the future will see such systems playing an important role in implementing presently available and future more general reliability analyses.

References

9-1. Katz, D. L.; et. al.: Computers in Engineering Design Education, Vol. I - Summary. The University of Michigan, College of Engineering, April 1, 1966.

9-2. Fenves, S. J.; et. al.: STRESS-A User's Manual. MIT Press, Cambridge, Mass., 1964.

9-3. IBM Publication H20-0170-1. 1620 Electronic Circuit Analysis Program (ECAP) (1620-EE-02X) User's Manual. Tech. Publications Dept., White Plains, N. Y., 1965.

9-4. Computer-Aided Circuit Design Seminar Proceedings. Kresge Auditorium, MIT, Cambridge, Mass., April 11-12, 1967, NASA TMX-59610.

9-5. So, H. C.: OLCA - An On-line Circuit Analysis System. Published in Computer-Aided Circuit Design Seminar Proceedings. Kresge Auditorium, MIT, Cambridge, Mass., April 11-12, 1967, pp. 9-11, NASA TMX-59610.

9-6. Hogsett, G. R.; Nisewanger, D. A.; and O'Hara, A. C., Jr.: Application Experiment with On-line Graphics-Aided ECAP. 1967 International Solid-State Circuits Conference Digest of Technical Papers, University of Penn., Philadelphia, Pa., Feb. 1967, pp. 72-73.

Appendix A

Revised Version of PVA Program Listed in Ref. 4-3

```
      SOURCE LIST
C  **** PERFORMANCE ANALYSIS PROGRAM ****
      DIMENSION HI(5),NV(5),J1(5),ALPHA(20),SY1(5),SY2(5),SY3(5),SY4(5)
     1,SCP(5,5),TM(20),ISD(20),HD(20),A(3,10),IRAND(20),RHO(20,20),R(20,
     220),UR(2),D(20),RN(20),RS(20),RSS(20,20),B(100),Y(100,5),AMU1(5),A
     3MU2(5),AMU3(5),AMU4(5),SIG(5),GAM1(5),GAM2(5),STD(5),Z(5,13),ELPH(
     45,13),AH(2,3),NE1(20),NE2(20)
      COMMON UR,XN,LOOP,Z,GAM1,GAM2,AMU1,AMU2,SIG,ELPH
C
C       INPUT GENERAL INFORMATION
C
C        XN . . . . STARTING VALUE FOR RANDOM NUMBER GENERATOR
C        NM . . . . NUMBER OF MODEL
C        HI . . . . MODEL NAMES (ALPHANUMERIC)
C        NV . . . . NUMBER OF VARIABLE IN MODEL
C        J1 . . . . NUMBER OF CORRELATED VARIABLES IN MODEL
C        LIM1 . . . NUMBER OF DATA POINTS TO BE GENERATED
C        A. . . . . SUBROUTINE NAMES (ALPHANUMERIC)
C        AH . . . . DISTRIBUTION NAMES (ALPHANUMERIC)
C        TM . . . . NOMINAL VALUES
C        TSD. . . . DEVIATION VALUES
C        HD . . . . VARIABLE NAMES (ALPHANUMERIC)
C        IRAND . . RANDOM NUMBER CALL VALUE
C        NE . . . . PARAMETERS FOR BETA AND GAMMA DISTRIBUTIONS
C        ALPHA. . . PARAMETER FOR WEIBULL VARIABLES
C        RHO. . . . INPUT CORRELATIONS
C
      READ 98,XN
      READ 50,NM,(HI(I),I=1,NM)
      READ 51,(NV(I),J1(I),I=1,NM),LIM1
      READ 99,((A(I,J),I=1,3),J=1,10)
      READ 99,((AH(I,J),J=1,3),I=1,2)
       AN=LIM1
      LINE = 0
      LOOP=0
      DO 1 I=1,NM
      SY1(I)=0.
      SY2(I)=0.
      SY3(I)=0.
      SY4(I)=0.
      DO 1 J=1,NM
      SCP(I,J)=0.
    1 CONTINUE
      DO 31 I=1,NM
      K=NV(I)
      DO 100J=1,K
      RS(J)=0.
      DO 100M=1,K
      R(J,M)=0.
      RHO(J,M)=0.
      RSS(J,M)=0.
  100 CONTINUE
C
C       INPUT NOMINAL AND DEVIATION VALUES
C
      READ 53,(TM(J),ISD(J),HD(J),IRAND(J),NE1(J),NE2(J),ALPHA(J),J=1,K)
      PRINT 62
      PRINT 54,I,HI(I)
```

```
      DO 2 J=1,K
      M=IRAND(J)
      PRINT 55,J,HD(J),TM(J),TSD(J),A(1,M),A(2,M),A(3,M)
    2 CONTINUE
      J2=J1(I)
      IF(J1(I))5,5,3
    3 J2=J1(I)
C
C       INPUT CORRELATIONS
C
      READ 56,((RHO(N,M),M=N,J2),N=1,J2)
      PRINT 57
      DO 4 MM=2,J2
      M=MM-1
      PRINT 58,(RHO(N,MM),N=1,M)
    4 CONTINUE
C
C       TRANSFORM CORRELATION MATRIX
C
      CALL SQRM(RHO,J2,K)
    5 L=0
      PRINT 59,(HD(M),M=1,K),HI(I)
C
C       CHOOSE RANDOM DISTRIBUTION SUBROUTINE
C         AND CALCULATE PARAMETER VALUES
C
    6 L=L+1
      DO 16 J=1,K
      IR=IRAND(J)
      GO TO(7,8,9,10,11,12,13,14),IR
    7 CALL UNIFM(1)
      ARG = TM(J) + (4.0*UR(1)-2.0)*TSD(J)
      GO TO 15
    8 CALL NORM(ARG)
      ARG = ARG * TSD(J) + TM(J)
      GO TO 15
C     CALL FOR LOGNORMAL SUBROUTINE
    9 CALL  NORM(ARG)
      ARG = EXP(ARG*TSD(J)+TM(J))
      GO TO 15
   10 CALL EXPN(TSD(J),ARG)
      GO TO 15
   11 CALL WEIB(TSD(J),ALPHA(J),ARG)
      GO TO 15
   12 CALL GAMMA(TSD(J),NE1(J),ARG)
      GO TO 15
   13 CALL BETA(ALPHA(J),NE1(J),NE2(J),ARG)
      ARG = ARG*TSD(J)*4.0 - TSD(J)*2.0 + TM(J)
      GO TO 15
   14 CALL CHISQ(TSD(J),NE1(J),ARG)
   15 RN(J)=ARG
   16 CONTINUE
      IF(J1(I))20,20,17
   17 DO 18J=1,J2
      D(J)=0.
      DO 18M=1,J
      D(J)=D(J)+RN(M)*R(M,J)
   18 CONTINUE
      DO 19 J=1,J2
```

```
      RN(J)=D(J)
   19 CONTINUE
C
C       CALCULATE INPUT CHECK
C
   20 CONTINUE
      DO 22 J=1,K
      RS(J)=RS(J)+RN(J)
      DO 22 M=1,K
      RSS(J,M)=RSS(J,M)+RN(J)*RN(M)
   22 CONTINUE
      DO 123 J=1,K
      X(J) = RN(J)
  123 CONTINUE
      CALL MODEL(RN,I,Y(L,I))
      PRINT 60,(RN(M),M=1,K),Y(L,I)
      LINE = LINE + (K+16)/8
       IF(LINE-44)310,300,300
  300 PRINT 320
      LINE = 0
  310 IF(LIM1-L)23,23,6
   23 CONTINUE
      LINE = 0
      DO 24 J=1,K
      RSS(J,J)=SQRT((RSS(J,J)-RS(J)*RS(J)/AN)/(AN-1.))
      RS(J)=RS(J)/AN
   24 CONTINUE
      PRINT 61
      PRINT 52,I,HI(I)
      DO 25 J=1,K
      M=IRAND(J)
      PRINT 55,J,HD(J),RS(J),RSS(J,J),A(1,M),A(2,M),A(3,M)
   25 CONTINUE
      IF(J2)31,31,26
   26 DO 28J=1,J2
      DO 28M=1,J2
      IF(J-M)27,28,27
   27 RSS(J,M)=((RSS(J,M)-RS(J)*RS(M)*AN)/(AN-1.))/(RSS(J,J)*RSS(M,M))
   28 CONTINUE
      DO 29 J=1,J2
      RSS(J,J)=1.
   29 CONTINUE
      PRINT 57
      DO 30 JJ=2,J2
      J=JJ-1
       PRINT 58,(RSS(JJ,M),M=1,J)
   30 CONTINUE
   31 CONTINUE
C
C       ARRANGE DEPENDENT DATA IN ASSENDING ORDER
C
      DO39N=1,NM
      KK=1
      B(1)=Y(1,N)
      DO 37 I=2,LIM1
      IF(B(KK)-Y(I,N))32,32,34
   32 B(I)=Y(I,N)
   33 KK=I
      GO TO 37
```

```
   34 DO 35 M=1,KK
      J=I-M
      IF(B(J)-Y(I,N))36,36,35
   35 B(J+1)=B(J)
      B(J)=Y(I,N)
      GO TO 33
   36 B(J+1)=Y(I,N)
      KK=I
   37 CONTINUE
      DO 38 I=1,LIM1
      Y(I,N)=B(I)
   38 CONTINUE
   39 CONTINUE
      PRINT 63,(HI(I),I=1,NM)
      DO 40 I=1,LIM1
      PER=FLOAT(I)/AN
      LINE=LINE+1
      IF(LINE-44)340,340,330
  330 PRINT 320
      LINE = 0
  340 PRINT 64,I,PER,(Y(I,N),N=1,NM)
   40 CONTINUE
      DO 42 N=1,LIM1
      DO 41 I=1,NM
      TY=Y(N,I)
      SY1(I)=SY1(I)+TY
      YT=TY*TY
      SY2(I)=SY2(I)+YT
      SY3(I)=SY3(I)+YT*TY
      SY4(I)=SY4(I)+YT*YT
      DO 41 J=1,NM
      SCP(I,J)=SCP(I,J)+Y(N,I)*Y(N,J)
   41 CONTINUE
   42 CONTINUE
C
C       CALCULATE DISTRIBUTION MOMENTS
C
      DO 43 I=1,NM
      AMU1(I)=SY1(I)/AN
      AMU2(I)=0.
      AMU3(I)=0.
      AMU4(I)=0.
      DO420N=1,LIM1
      YC=Y(N,I)-AMU1(I)
      YCSQ=YC*YC
      AMU2(I)=AMU2(I)+YCSQ
      AMU3(I)=AMU3(I)+YCSQ*YC
      AMU4(I)=AMU4(I)+YCSQ*YCSQ
  420 CONTINUE
      SIG(I)=SQRT(AMU2(I)/AN)
      GAM1(I)=AMU3(I)/(SIG(I)*AMU2(I))
      GAM2(I)=AMU4(I)/(AMU2(I)*AMU2(I))
      STD(I)=SQRT(AMU2(I)/(AN-1.))
   43 CONTINUE
      DO 44 I=1,NM
      DO 44 J=I,NM
      SCP(I,J)=(SCP(I,J)-AMU1(I)*AMU1(J)*AN)/(AN-1.)
   44 CONTINUE
      PRINT 65,(HI(I),I=1,NM)
```

```
      PRINT 66,(AMU1(I),I=1,NM)
      PRINT 67,(AMU2(I),I=1,NM)
      PRINT 68,(AMU3(I),I=1,NM)
      PRINT 69,(AMU4(I),I=1,NM)
      PRINT 70,(STD(I),I=1,NM)
      PRINT 71,(GAM1(I),I=1,NM)
      PRINT 72,(GAM2(I),I=1,NM)
      PRINT 73,NM
      DO 45 I=1,NM
      PRINT 74,HI(I),(SCP(I,J),J=1,NM)
   45 CONTINUE
      DO49I=1,NM
      Z(I,1)=AMU1(I)-3.*STD(I)
      DO146J=2,13
      Z(I,J)=Z(I,J-1)+0.5*STD(I)
  146 CONTINUE
C
C      CHOOSE SUBROUTINE TO FIT DISTRIBUTION OF MODEL
C
      IF(GAM1(I)-0.5)46,46,47
   46 CONTINUE
      CALL EDGE(I)
      LOP=1
      GO TO 48
   47 CONTINUE
      CALL LAGUR(I,AN,SY2(I),SY3(I),SY4(I))
      LOP=2
   48 PRINT 75,HI(I),(AH(LOP,J),J=1,3)
      PRINT 76,(Z(I,J),ELPH(I,J),J=1,13)
   49 CONTINUE
      PUNCH 98,XN
   50 FORMAT(I2,5A4)
   51 FORMAT(11I5)
   52 FORMAT (6H0MODEL,I3,2H, ,A4,10X,10HVAR. NAMES,5X,13H     MEANS
     15X,9HSTD. DEV.,5X,12HDISTRIBUTION)
   53 FORMAT(2E10.4,A4,1X,I3,2I2,E10.4)
   54 FORMAT (6H0MODEL,I3,2H, ,A4,10X,10HVAR. NAMES,5X,13HNOMINAL VALUE,
     15X,9HDEVIATION,5X,12HDISTRIBUTION)
   55 FORMAT(19X,I3,6X,A4,9X,E12.5,6X,E11.5,4X,3A4)
   56 FORMAT(16F5.0)
   57 FORMAT(19H0INPUT CORRELATIONS//)
   58 FORMAT(1H ,20F5.3)
   59 FORMAT(1H-,5X,8(5X,A4,3X)/3X,8(5X,A4,3X)/8(5X,A4,3X))
   60 FORMAT(1H0,5X,8E12.4/3X,8E12.4/8E12.4)
   61 FORMAT(12H-INPUT CHECK)
   62 FORMAT(1H-)
   63 FORMAT(41H-DEPENDENT DATA LISTED IN ASCENDING ORDER,//4H    I,
     15X,5HI/N  ,5(7X,A4,3X))
   64 FORMAT(I4,F10.3, 5E14.4)
   65 FORMAT(8H-MOMENTS/10X,5(7X,A4,4X))
   66 FORMAT(10H0     FIRST,5E15.6)
   67 FORMAT(10H0    SECOND,5E15.6)
   68 FORMAT(10H0     THIRD,5E15.6)
   69 FORMAT(10H0    FOURTH,5E15.6)
   70 FORMAT(10H0STD. DEV.,5E15.6)
   71 FORMAT(10H0 SKEWNESS,5E15.6)
   72 FORMAT(10H0 KURTOSIS,5E15.6)
   73 FORMAT(36H0VARIANCE - COVARIANCE MATRIX, ORDER,I2)
   74 FORMAT(1H0,3X,A4,2X,5E15.6)
```

```
 75 FORMAT(23H-PERCENTAGE POINTS FOR ,A4,4H BY ,3A4)
 76 FORMAT(5H0 Z =,F10.5,10H     F(Z) =,E13.5)
 98 FORMAT(F10.0)
 99 FORMAT(20A4)
320 FORMAT(1H-//)
    STOP
    END

    SOURCE LIST
    SUBROUTINE SQRT(RHO,N,R)
    DIMENSION RHO(20,20),R(20,20)
    DO 8 I=1,N
    DO 8 J=I,N
    KK=1
    P=RHO(I,J)
  1 IF(KK-I)2,3,3
  2 P=P-R(KK,J)*R(KK,I)
    KK=KK+1
    GO TO 1
  3 IF(J-I)8,4,7
  4 IF(P)5,6,6
  5 PRINT 10,I,J,R(I,J)
  6 R(I,J)=SQRT(P)
    GO TO 8
  7 R(I,J)=P/R(I,I)
  8 CONTINUE
    RETURN
 10 FORMAT(9H-ELEMENT 2I3,12HIS EQUAL TO ,E15.6)
    END

    SOURCE LIST
    SUBROUTINE EDGE(J)
    DIMENSION UR(2),Z(5,13),GAM1(5),GAM2(5),AMU1(5),AMU2(5),SIG(5),
   1ELPH(5,13)
    COMMON UR,XN,LOOP,Z,GAM1,GAM2,AMU1,AMU2,SIG,ELPH
    Y1 = .14112821
    Y2 = .08864027
    Y3 = .02743349
    Y4 = .00039446
    Y5 = .00328975
    DO 3 I=1,13
    ZR = (Z(J,I)-AMU1(J))/SIG(J)
    Z2 = ZR * ZR
    Z3 = ZR * Z2
    ZP = ABS(ZR)/1.41422
    ZP2 = ZP * ZP
    ZP3 = ZP * ZP2
    DENOM = (1.+Y1*ZP+Y2*ZP2+Y3*ZP3+Y4*ZP2*ZP2+Y5*ZP2*ZP3)**8
    TERM1 = 0.5 * (1.-1./DENOM)
    TERM2 = 0.3989* EXP(-Z2*0.5)*((-GAM1(J)/6.)*(Z2-1.)
   1+(GAM2(J)-3.)/24.*(3.*ZR-Z3)+GAM1(J)*GAM1(J)
   2*(10.*Z3-Z2*Z3-15.*ZR)/72.)
    IF(ZR)1,1,2
  1 ELPH(J,I) = 0.5-TERM1+TERM2
    GO TO 3
  2 ELPH(J,I)= 0.5+TERM1+TERM2
  3 CONTINUE
    RETURN
    END
```

```
   SOURCE LIST
   SUBROUTINE LAGUR(J,AN,SY2,SY3,SY4)
   DIMENSION UR(2),Z(5,13),GAM1(5),GAM2(5),AMU1(5),AMU2(5),SIG(5),
  1ELPH(5,13)
   COMMON UR,XN,LOOP,Z,GAM1,GAM2,AMU1,AMU2,SIG,ELPH
   ALP = AMU1(J)/AMU2(J)
   ALM = AMU1(J) * ALP
   LAMDA = ALM
   AMD = LAMDA
   TEST = 2.*(ALM-AMD)
   IF(TEST-1.)1,2,2
 1 IF(AMD)3,2,3
 2 AMD=AMD+1.
   LAMDA=LAMDA+1
 3 LAM=LAMDA-1
   AL2=ALP*ALP
   DEN1=(AMD+1.)*(AMD+2.)
   DEN2=DEN1*(AMD+3.)
   DEN3=DEN2*AMD
   V2=SY2 / AN + AMU1(J)*AMU1(J)
   B1=(AMU1(J)*ALP-AMD)/AMD
   B2=(V2*AL2-2.*(AMD+1.)*AMU1(J)*ALP+AMD*(AMD+1.))/(2.*AMD*(AMD+1.
  1))
   B3=((SY3/AN)*ALP*AL2-3.*(AMD+2.)*V2*AL2+3.*DEN1*AMU1(J)
  1*ALP-AMD*DEN1)/(6.*AMD*DEN1)
   B4=((SY4/AN)*AL2*AL2-4.*(AMD+3.)*(SY3/AN)*ALP
  1*AL2+6.*(AMD+2.)*(AMD+3.)*V2*AL2-4.*DEN2*AMU1(J)
  2*ALP+DEN3)/(24.*DEN3)
   DO 8 I=1,13
   X=Z(I)*ALP
   IF(LAM)4,4,5
 4 TERM1=-1.
   COE=1.
   GO TO 7
 5 COE=AMD-1.
   TERM1=-X**LAM
   DO 6 K=1,LAM
   TERM = TERM1-COE*(X**(LAM-K))
   IF(K-LAM)6,7,7
   COE=COE*(AMD-(FLOAT(K)+1.))
 6 CONTINUE
 7 TERM2=-B1+B2*(-X+AMD+1.)+B3*(-X*X+2.*(AMD+2.)
  1*X-DEN1)+B4* (-X**3  +3.*(AMD+3.)*X*X-3.*
  2(AMD+3.)*(AMD+2.)*X+DEN2)
   ELPH(J,I)=1.+EXP(-X)*(TERM1+(X**LAMDA)*TERM2)/COE
 8 CONTINUE
   RETURN
   END

   SOURCE LIST
   SUBROUTINE UNIFM(N)
   DIMENSION UR(2),Z(5,13),GAM1(5),GAM2(5),AMU1(5),AMU2(5),SIG(5),
  1ELPH(5,13)
   COMMON UR,XN,LOOP,Z,GAM1,GAM2,AMU1,AMU2,SIG,ELPH
   DO 1 I=1,N
   RC=33.*XN+101.
   XP=RC/2048.
   MU=XP
```

```
      UM=MU
      UR(I)=XP-UM
      XN=RC-UM*2048.
    1 CONTINUE
      RETURN
      END

      SOURCE LIST
      SUBROUTINE NORM(ONE)
      DIMENSION UR(2),Z(5,13),GAM1(5),GAM2(5),AMU1(5),AMU2(5),SIG(5),
     1ELPH(5,13)
      COMMON UR,XN,LOOP,Z,GAM1,GAM2,AMU1,AMU2,SIG,ELPH
      IF(LOOP)1,1,2
    1 CALL UNIFM(2)
      GS=-2.*ALOG(UR(1))
      GS=SQRT(GS)
      H=6.283185*UR(2)
      ONE=GS*COS(H)
      TWO=GS*SIN(H)
      LOOP=1
      RETURN
    2 ONE=TWO
      LOOP=0
      RETURN
      END

      SOURCE LIST
      SUBROUTINE EXPN (THETA,ARG)
      DIMENSION UR(2),Z(5,13),GAM1(5),GAM2(5),AMU1(5),AMU2(5),SIG(5),
     1ELPH(5,13)
      COMMON UR,XN,LOOP,Z,GAM1,GAM2,AMU1,AMU2,SIG,ELPH
      CALL UNIFM(1)
      ARG=-ALOG(UR(1))*THETA
      RETURN
      END

      SOURCE LIST
      SUBROUTINE WEIB(THETA,ALPHA,ARG)
      DIMENSION UR(2),Z(5,13),GAM1(5),GAM2(5),AMU1(5),AMU2(5),SIG(5),
     1ELPH(5,13)
      COMMON UR,XN,LOOP,Z,GAM1,GAM2,AMU1,AMU2,SIG,ELPH
      CALL EXPN(THETA,ARG1)
      ARG = ARG1 * * (1./ALPHA)
      RETURN
      END

      SOURCE LIST
      SUBROUTINE GAMMA(THETA,N,ARG)
      DIMENSION UR(2),Z(5,13),GAM1(5),GAM2(5),AMU1(5),AMU2(5),SIG(5),
     1ELPH(5,13)
      COMMON UR,XN,LOOP,Z,GAM1,GAM2,AMU1,AMU2,SIG,ELPH
      G=0.
      DO 1 I=1,N
      CALL UNIFM(1)
      G=G+ALOG(UR(1))
    1 CONTINUE
      ARG = -G*THETA
      RETURN
      END
```

```
      SOURCE LIST
      SUBROUTINE BETA(ALPHA,N1,N2,ARG)
      DIMENSION UR(2),Z(5,13),GAM1(5),GAM2(5),AMU1(5),AMU2(5),SIG(5),
     1ELPH(5,13)
      COMMON UR,XN,LOOP,Z,GAM1,GAM2,AMU1,AMU2,SIG,ELPH
      CALL GAMMA(ALPHA,N1,ARG1)
      CALL GAMMA(ALPHA,N2,ARG2)
      ARG = ARG1/(ARG1+ARG2)
      RETURN
      END

      SOURCE LIST
      SUBROUTINE CHISQ(THETA,NDF,ARG)
      DIMENSION UR(2),Z(5,13),GAM1(5),GAM2(5),AMU1(5),AMU2(5),SIG(5),
     1ELPH(5,13)
      COMMON UR,XN,LOOP,Z,GAM1,GAM2,AMU1,AMU2,SIG,ELPH
      ARG=0.
      DO 1 I=1,NDF
      CALL NORM(ARG1)
      ARG = ARG + ARG1 * ARG1
    1 CONTINUE
      ARG = ARG * THETA
      RETURN
      END

      SOURCE LIST
      SUBROUTINE MODEL(X,J,Y)
C     SUBROUTINE FOR FUNCTIONAL FORM OF PERFORMANCE ATTRIBUTES
      DIMENSION UR(2),Z(5,13),GAM1(5),GAM2(5),AMU1(5),AMU2(5),SIG(5),
     1ELPH(5,13),X(20)
      COMMON UR,XN,LOOP,Z,GAM1,GAM2,AMU1,AMU2,SIG,ELPH
      X(10) = 0.003
      Y = X(1)/X(10)**(0.74)+X(2)/X(10)**(0.528)+8760.*X(10)*(X(3)/X(10)
     1**(0.528)+X(4)/SQRT(X(10)))+X(5)+X(6)+X(7)+8760.*X(10)*(X(8)+X(9))
      RETURN
      END
```

Appendix B

Bounds for Reliability Program

```
C      ***  BOUNDS FOR RELIABILITY      ***
C       *    ANALYSIS OF PATHS AND CUTS  *
       DIMENSION IP(100,20),IC(20,20)
       DIMENSION IUN(20),IB(20),BOUND(20),PROB(20),IO(10)
     1 READ 1500,N
       READ 1510, (PROB(I),I=1,N), EPSLON
       PRINT 1000,N,(I,PROB(I),I=1,N)
       CALL PATH(N,NP,IP)
       PRINT 1010,NP
       DO 450 I=1,NP
       DO 400 J=1,20
   400 IUN(J) = 0
       DO 430J=1,20
       K = IP(I,J)
       IF( K )430,430,410
   410 IF(K-25)420,430,430
   420 IUN(K) = 1
   430 CONTINUE
       DO 440 J=1,20
   440 IP(I,J) = IUN(J)
   450 CONTINUE
       DO 4 I=1,NP
       K=0
       DO 3 J=1,N
       IF(IP(I,J))3,3,2
     2 K=K+1
       IO(K)=J
     3 CONTINUE
       PRINT 1020,I,(IO(J),J=1,K)
     4 CONTINUE
C    ***    DETERMINE SYSTEM CUTS
C    ***    CHECK FOR SINGLE ELEMENT CUTS
       K=1
       DO 30 I=1,N
       DO 10 J=1,NP
       IF(IP(I,J))10,10,30
    10 CONTINUE
    20 IC(K,I)=1
       K=K+1
    30 CONTINUE
C     ***    CHECK FOR DOUBLE ELEMENT CUTS
       N1=N-1
       DO 90 I=1,N1
       I1=I+1
       DO 90 J=I1,N
       IDUM=0
       DO 40 L=1,NP
       ITRICK=IP(L,I).OR.IP(L,J)
       IDUM=IDUM+ITRICK
    40 CONTINUE
       IF(IDUM-NP) 90,50,90
    50 IC(K,I)=IC(K,J)=1
       K1=K-1
       DO 70 L=1,K1
       DO 60 M=1,N
       IDUM=IC(K,M).AND.IC(L,M)
       IF(IDUM)60,60,80
    60 CONTINUE
```

```
   70 CONTINUE
      K=K+1
      GO TO 90
   80 IC(K,I)=IC(K,J)=0
   90 CONTINUE
C    ***   CHECK FOR TRIPLE ELEMENT CUTS THAT ARE MINIMAL
      N2=N-2
      DO 180 I=1,N2
      I1=I+1
      DO 180 J=I1,N1
      I2=J+1
      DO 180 L=I2,N
      IDUM=0
      DO 120 M=1,NP
      ITRICK=(IP(M,I).OR.IP(M,J)).OR.IP(M,L)
      IDUM=IDUM+ITRICK
  120 CONTINUE
      IF(IDUM-NP)180,130,180
  130 DO 135 II=1,N
  135 IC(K,II) = 0
      IC(K,I)=IC(K,J)=IC(K,L)=1
      K1=K-1
      DO 170 M=1,K1
      IDUM=JDUM=0
      DO 140 IJ=1,N
      ITRICK=IC(M,IJ).AND.IC(K,IJ)
      IDUM=IDUM+ITRICK
      JDUM=JDUM+IC(M,IJ)
  140 CONTINUE
      IF(IDUM-JDUM)170,150,170
  150 IC(K,I)=IC(K,J)=IC(K,L)=0
      GO TO 180
  170 CONTINUE
      K = K + 1
  180 CONTINUE
C *** CHECK FOR FOUR ELEMENT CUTS
      N3=N-3
      DO 510 I=1,N3
      I1=I+1
      DO 510 J=I1,N2
      I2=J+1
      DO 510 KK=I2,N1
      I3=KK+1
      DO 510 L=I3,N
      IDUM=0
      DO 460 M=1,NP
      ITRICK=(IP(M,I).OR.IP(M,J)).OR.(IP(M,KK).OR.IP(M,L))
      IDUM=IDUM+ITRICK
  460 CONTINUE
      IF(IDUM-NP)510,470,510
  470 IC(K,I)=IC(K,J)=IC(K,KK)=IC(K,L)=1
      K1=K-1
      DO 500 M=1,K1
      IDUM=JDUM=0
      DO 480 IJ=1,N
      ITRICK=IC(M,IJ).AND.IC(K,IJ)
      IDUM=IDUM+ITRICK
      JDUM=JDUM+IC(M,IJ)
  480 CONTINUE
```

```
      IF(IDUM-JDUM)500,490,500
  490 IC(K,I)=IC(K,J)=IC(K,KK)=IC(K,L)=0
      GO TO 510
  500 CONTINUE
      K=K+1
  510 CONTINUE
C *** CHECK FOR FIVE ELEMENT CUTS
      N4=N-4
      DO 570 I=1,N4
      I1=I+1
      DO 570 J=I1,N3
      I2=J+1
      DO 570 KK=I2,N2
      I3=KK+1
      DO 570 L=I3,N1
      I4=L+1
      DO 570 MM=I4,N
      IDUM=0
      DO 520 M=1,NP
      ITRICK=((IP(M,I).OR.IP(M,J)).OR.IP(M,KK)).OR.(IP(M,L).OR.IP(M,MM))
      IDUM=IDUM+ITRICK
  520 CONTINUE
      IF(IDUM-NP)570,530,570
  530 IC(K,I)=IC(K,J)=IC(K,KK)=IC(K,L)=IC(K,MM)=1
      K1=K-1
      DO 560 M=1,K1
      IDUM=JDUM=0
      DO 540 IJ=1,N
      ITRICK=IC(M,IJ).AND.IC(K,IJ)
      IDUM=IDUM+ITRICK
      JDUM=JDUM+IC(M,IJ)
  540 CONTINUE
      IF(IDUM-JDUM)560,550,560
  550 IC(K,I)=IC(K,J)=IC(K,KK)=IC(K,L)=IC(K,MM)=0
      GO TO 570
  560 CONTINUE
      K=K+1
  570 CONTINUE
C   ***   ALL CUTS HAVE BEEN DETERMINED   ***
      NC = K - 1
      PRINT 1030,NC
      DO 195 I=1,NC
      K=0
      DO 190 J=1,N
      IF(IC(I,J))190,190,185
  185 K=K+1
      IO(K)=J
  190 CONTINUE
      PRINT 1020,I,(IO(J),J=1,K)
  195 CONTINUE
      SYSBD = 1.0
      DO 370 I=1,NC
      IM1 = I-1
      DO 200 K=1,IM1
  200 IB(K) = K
      NM = INCOE(NC,I)
      IJ = IM1
      BOUND(I) = 0.0
      DO 320 J=1,NM
```

```
      K = 1
      IF(IJ-NC)250,210,210
  210 IK = I-K
      IF(IB(IK)-(NC-K))230,220,220
  220 K = K+1
      GO TO 210
  230 IB(IK) = IB(IK) + 1
      DO 240 JI=IK,I
  240 IB(JI+1) = IB(JI) + 1
      IJ = IB(I-1) + 1
      GO TO 260
  250 IJ = IJ + 1
  260 DO 270 JI = 1,N
  270 IUN(JI) = IC(IJ,JI)
      DO 290 JI=1,IM1
      N1 = IB(JI)
      DO 280 IK=1,N
  280 IUN(IK) = IUN(IK).OR.IC(N1,IK)
  290 CONTINUE
      PR = 1.0
C   ***  CALCULATE EVENT PROBABILITY
      DO 310 IK=1,N
      IF(IUN(IK))310,310,300
  300 PR = PR * (1.0-PROB(IK))
  310 CONTINUE
      BOUND(I) = BOUND(I) + PR
  320 CONTINUE
C   ***  PRINT BOUNDS (UPPER OR LOWER) AND TEST TO DETERMIND CONVERGENTS
C   ***                          OF SERIES CALCULATIONS
      II = MOD(I,2) + 1
      SYSBD = SYSBD + BOUND(I)*(-1.0)**(II-1)
      GO TO (340, 330), II
  330 PRINT 2010,SYSBD,BOUND(I)
      GO TO 350
  340 PRINT 2020,SYSBD,BOUND(I)
  350 IF(I-1)370,370,360
  360 IF(ABS(BOUND(I)-BOUND(I-1))-EPSLON)380,380,370
  370 CONTINUE
      PRINT 2030,SYSBD
  380 GO TO 1
 1000 FORMAT(57H-  B O U N D S  F O R  S Y S T E M  R E L I A B I L I T
     1Y//7X,16HCIRCUIT CONTAINS,I3,9H ELEMENTS//11X,7HELEMENT,15X,11HPRO
     2BABILITY/11X,7HNUMBER,16X,11HOF  SUCCESS//(I15,F26.4))
 1010 FORMAT(///7X,27HTIE SETS OR SUCCESS PATHS (,I3,2H )//12X,4HPATH,
     15X,15HELEMENT NUMBERS/)
 1020 FORMAT(I15,8X,16I5)
 1030 FORMAT(///7X, 9HCUT SETS(,I3,2H ))
 2010 FORMAT(16H0LOWER BOUND IS ,E11.5,5X,10HLAST TERM E11.5)
 2020 FORMAT(16H0UPPER BOUND IS ,E11.5,5X,10HLAST TERM E11.5)
 2030 FORMAT(20H0SYSTEM RELIABILITY E12.5)
 1500 FORMAT(10I5)
 1510 FORMAT(8E10.4)
      END

      FUNCTION INCUE(N,M)
      AN = N
      AM = M
      ALN = ALOG(AN)
      ALM = ALOG(AM)
```

```
      NM = M-1
      DO 10 K=1,NM
      AK = K
      ALN = ALN + ALOG(AN-AK)
      ALM = ALM + ALOG(AM-AK)
   10 CONTINUE
      TERM = EXP(ALN-ALM)
      INCOE = TERM + 0.1
      RETURN
      END

      SUBROUTINE PATH(NEQP,IMAX,L)
      DIMENSION ITABLE( 25,9),IPRED(9),L(100,20)
      DO 100 I=1,20
      DO 100 J=1,20
  100 L(I,J)=0
      JB=NEQP+1
      DO 4 I=1,JB
      READ 2, IACTIV,IPRED
      DO 4 J=1,9
    4 ITABLE(IACTIV,J)=IPRED(J)
      J=1
      IMAX=1
      L(1,1)= 25
    6 J=J+1
      IC=0
      ICOUNT=0
      DO 12 I=1,IMAX
      IF(L(I,J-1))7,7,8
    7 IC=IC+1
      GO TO 12
    8 K=L(I,J-1)
      M=1
      L(I,J)=ITABLE(K,M)
    9 M=M+1
      IF(ITABLE(K,M))10,12,10
   10 ICOUNT=ICOUNT+1
      KBC=IMAX+ICOUNT
      DO 11 KK=1,J
   11 L(KBC,KK)=L(I,KK)
      L(KBC,J)=ITABLE(K,M)
      GO TO 9
   12 CONTINUE
      IF(IC-IMAX)13,14,14
   13 IMAX=IMAX+ICOUNT
      GO TO 6
   14 RETURN
    2 FORMAT(10I5)
      END
```

Appendix C

Reliability Cost Trade-Off Analysis Program

```
C
C     RELIABILITY COST TRADE-OFF ANALYSIS
C
C     THIS PROGRAM CALCULATES SYSTEM RELIABILITY FOR
C     SYSTEMS OF ELEMENTS HAVING THE FOLLOWING PROPERTIES
C        NEL              NUMBER OF ELEMENTS
C        N                IDENTICAL ITEMS IN PARALLEL FOR EACH ELEMENT
C        NO               MINIMUM ITEMS NECESSARY FOR OPERATION
C        IS               IDENTICAL SPARES
C        IR               IDENTICAL STANDBY REDUNDANCIES
C
C     INPUT VARIABLES IN THE SYSTEM ARE
C        TIME(I)          TIME OR LENGTH OF MISSION
C        FRATE(I)         FAILURE RATE
C        RELSW(I)         SWITCH RELIABILITY
C        SWCST(I)         SWITCH COST
C        ELCST(I)         ACTIVE ITEM COST
C        SPCST(I)         SPARE ITEM COST
C        RSCST(I)         STANDBY ITEM COST
C
C     OTHER INPUT VARIABLES NECESSARY FOR INITIALIZATION
C        IND(I,J)         THESE ARE INDICATORS OF ITEMS ALLOWED TO VARY
C        INPRM(I,J)       THESE ARE THE INITIAL NUMBERS OF ITEMS IN THE
C                         SYSTEM
C
C     DEFINITIONS OF OTHER VARIABLES USED IN MAIN ROUTINE
C        CONVG            CONVERGENCE CRITERION FOR HALTING ANALYSIS
C        TLAM             MEAN NUMBER OF FAILURES
C        MS               NUMBER OF SPARES PLUS NUMBER OF STANDBYS
C        R(I)             ELEMENT RELIABILITY AT I-TH STEP
C        RBASE            MAXIMUM RELIABILITY OF THE SYSTEM AT I-TH STEP
C        CBASE            COST OF THE SYSTEM AT I-TH STEP
C        NPRM(I,J)        NUMBER OF ITEMS IN EACH ELEMENT AT I+1ST STEP
C        ELR(I,J)         ELEMENT RELIABILITY AT I+1ST STEP
C        REL(I,J)         SYSTEM RELIABILITY
C        CEL(I,J)         SYSTEM COST
C        RATIO(I,J)       RATIO OF RELIABILITY TO COST
C        RMAT             MAXIMUM RATIO, YEILDS RBASE AND CBASE.
C
      DIMENSION NPRM(10,3),IND(10,3),INPRM(10,3),REL(10,3),CEL(10,3),
     1RATIO(10,3),R(10),ELR(10,3)
      DIMENSION RELSW(10), TITLE(18)
      DIMENSION X(10),NO(10),TIME(10),FRATE(10),ELCST(10),SPCST(10),
     1SWCST(10),RSCST(10)
C
      COMMON X,NO,TIME,FRATE,RELSW,ELCST,SPCST,SWCST,RSCST,NEL,TLAM,MS,N
     1,NO
C
    1 READ 1000,NEL,CONVG
      READ 1003, TITLE
      DO 5 I=1,NEL
      READ 1001,          TIME(I),FRATE(I),RELSW(I),ELCST(I),
     1SPCST(I),SWCST(I),RSCST(I),NO(I)
      READ 1002,(IND(I,J),J=1,3),(INPRM(I,K),K=1,3)
    5 CONTINUE
C     SET UP INITIAL CONFIGURATION
      PRINT 1010,TITLE,NEL
      DO 20 I=1,NEL
```

```
      PRINT 1020,I,TIME(I),FRATE(I),RELSW(I),ELCST(I),
     1SPCST(I),RSCST(I),SWCST(I)
   20 CONTINUE
      PRINT 1030
      DO 25 I=1,NEL
      PRINT 1050,I,(INPRM(I,J),J=1,3),(IND(I,K),K=1,3)
   25 CONTINUE
      DO 10 I=1,NEL
      PRINT 1090
      CALL RELEST(I,INPRM,R(I))
   10 CONTINUE
      CALL MODEL(I,R,1.0,INPRM,RBASE,CBASE)
      PRINT 1060, RBASE, CBASE
      PRINT 1045
      PRINT 1040
      DO 26 I=1,NEL
      PRINT 1055,I,(INPRM(I,J),J=1,3)
      PRINT 1070,R(I)
   26 CONTINUE
      PRINT 1060,RBASE, CBASE
C     END OF INITIAL STATE
C
      DO 27 I=1,NEL
      DO 27 J=1,3
   27 NPRM(I,J) = INPRM(I,J)
      DO 40 I=1,NEL
      DO 40 J=1,3
      IF(IND(I,J)-1)40,30,31
   30 NPRM(I,J)=NPRM(I,J)+1
      PRINT 1090
      CALL RELEST(I,NPRM,ELR(I,J))
      NPRM(I,J)=INPRM(I,J)
      GO TO 40
   31 NPRM(I,J) = NPRM(I,J) + 2
      NO(I) = NO(I) + 1
      PRINT 1090
      CALL RELEST(I,NPRM,ELR(I,J))
      NPRM(I,J)=INPRM(I,J)
      NO(I) = NO(I) - 1
   40 CONTINUE
      L = 0
   41 DO 46 I=1,NEL
      DO 46 J=1,3
      IF(IND(I,J)-1)46,43,44
   43 NPRM(I,J)=NPRM(I,J)+1
      GO TO 45
   44 NPRM(I,J) = NPRM(I,J) + 2
   45 CONTINUE
      CALL MODEL(I,R,ELR(I,J), NPRM,REL(I,J),CEL(I,J))
      RATIO(I,J)=(REL(I,J)-RBASE)/((CEL(I,J)-CBASE)*RBASE)
      NPRM(I,J)=INPRM(I,J)
   46 CONTINUE
      RMAT=0.0
      DO 50 I=1,NEL
      DO 50 J=1,3
      IF(IND(I,J))50,50,48
   48 RMAT=AMAX1(RMAT,RATIO(I,J))
   50 CONTINUE
      DO 70 I=1,NEL
```

```
      DO 70 J=1,3
      IF(IND(I,J))70,70,55
   55 IF(RMAT-RATIO(I,J)) 70,60,70
   60 NI=I
      NJ=J
      GO TO 80
   70 CONTINUE
   80 IF(ABS(RBASE-REL(NI,NJ))-CONVG)110,110,85
   85 L = L + 1
      IF(IND(NI,NJ)-1)86,86,87
   86 INPRM(NI,NJ)=INPRM(NI,NJ)+1
      GO TO 88
   87 INPRM(NI,NJ) = INPRM(NI,NJ) + 2
      NO(NI) = NO(NI) + 1
   88 RBASE=REL(NI,NJ)
      CBASE=CEL(NI,NJ)
      R(NI)=ELR(NI,NJ)
C     OUTPUT NEW CONFIGURATION WITH RELIABILITY AND COST
      PRINT 1080,L
      PRINT 1040
      DO 100 I=1,NEL
      PRINT 1055,I,(INPRM(I,J),J=1,3)
      IF(I-NI)100,90,100
   90 PRINT 1070, R(I)
  100 CONTINUE
      PRINT 1060,RBASE,CBASE
      IF(IND(NI,NJ) - 1)101,101,102
  101 NPRM(NI,NJ) = INPRM(NI,NJ) + 1
      GO TO 105
  102 IF(INPRM(NI,NJ) - IND(NI,NJ))103,104,104
  103 NPRM(NI,NJ) = INPRM(NI,NJ) + 2
      NO(NI) = NO(NI) + 1
      GO TO 105
  104 IND(NI,NJ) = 0
      GO TO 106
  105 CALL RELEST(NI,NPRM,ELR(NI,NJ))
  106 NPRM(NI,NJ) = INPRM(NI,NJ)
      GO TO 41
  110 CONTINUE
      GO TO 1
      STOP
 1000 FORMAT(I5,E10.4)
 1001 FORMAT(7E10.0,I5)
 1002 FORMAT(6I5)
 1003 FORMAT(18A4)
 1010 FORMAT( 1H-//10X,43HRELIABILITY COST TRADE-OFF ANALYSIS (RECTA)//
     114X,18A4/14X,18HNUMBER OF ELEMENTS,I7//10X,
     217HINPUT INFORMATION/27X,20HFAILURE   SWITCH   *,9X,19HI T E M S
     3 C O S T,9X,1H*/10X,72HELEMENT   TIME    RATE     RELIAB.     ACTI
     4VE   SPARE   STANDBY   SWITCH/)
 1020 FORMAT(10X,I4,F10.0,F10.5,F10.5,2X,4F9.2)
 1030 FORMAT(1H0/10X,30HINITIAL  SYSTEM  CONFIGURATION /17X,65H*   I N I
     1 T I A L   I T E M S   * * *    I N D I C A T O R S    */10X,70HEL
     2EMENT  ACTIVE     SPARES     STANDBY   *   ACTIVE   SPARES   STAND
     3BY/)
 1040 FORMAT(10X,49HELEMENT  ACTIVE   SPARE   STANDBY     RELIABILITY/)
 1045 FORMAT(1H-/10X,12HINITIAL STEP//10X,21HSYSTEM  CONFIGURATION/17X,
     11H*9X,9HI T E M S,8X,1H*)
 1050 FORMAT(10X,I4,I9,I11,I12,I13,I9,I10)
```

```
1055 FORMAT(10X,I4,I9,I8,I9)
1060 FORMAT(1H0/10X,18HSYSTEM RELIABILITY,F10.6/10X,11HSYSTEM COST
    17X,F10.2)
1070 FORMAT(1H+,47X,F9.6)
1080 FORMAT(1H-/10X,4HSTEP,I3 //10X,21HSYSTEM  CONFIGURATION/17X,1H*,9X
    1,9HI T E M S,8X,1H*)
1090 FORMAT(1H-)
     END

     SUBROUTINE RELEST( IJ, INPRM, REL )
     DIMENSION INPRM(10,3)
     DIMENSION RELSW(10)
     DIMENSION X(10),NO(10),TIME(10),FRATE(10),ELCST(10),SPCST(10),
    1SWCST(10),RSCST(10)
     COMMON X,NO,TIME,FRATE,RELSW,ELCST,SPCST,SWCST,RSCST,NEL,TLAM,MS,N
    1,NO
     NO = NO(IJ)
     N = INPRM(IJ,1)
     M  = INPRM(IJ,2)
     IR = INPRM(IJ,3)
     PRB =  RELSW(IJ)
     T = TIME(IJ)
     ALAM = FRATE(IJ)
     PRINT 100,IJ,N,NO,M,IR,PRB,ALAM,T
     PRINT 110
     REL = 0.
     TLAM=T*ALAM
     DO 20 IS=0,IR
     IF( IR ) 4, 1, 4
   1 BIN = 1.0
     GO TO 16
   4 BINN=BIND=1.0
     IF(IS)5,15,5
   5 DO 10 I=1,IS
     BINN=BINN*FLOAT(IR-I+1)
     BIND=BIND*FLOAT(I)
  10 CONTINUE
  15 T = FLOAT(IR - IS)
     S = FLOAT(IS )
     QRB = 1.0 - PRB
     BIN = (BINN/BIND)*(PRB**S )*(QRB**T )
  16 MS=M+IS
     CALL RELPRB(R)
     PRINT 120, IS,R
     REL = REL+R * BIN
  20 CONTINUE
     PRINT 130, REL
     RETURN
 100 FORMAT(1H0/9X,34H RELIABILITY ESTIMATES FOR ELEMENT,I3,11H CONTAIN
    2ING//9X,30H   IDENTICAL ITEMS IN PARALLEL,20X,I4/9X,40H   MINIMUM
    3NUMBER OF ITEMS FOR OPERATION,10X,I4/9X,19H  IDENTICAL SPARES,31
    4X,I4/9X,40H   IDENTICAL ITEMS IN STANDBY REDUNDANCY,10X,I4/9X,21H
    5  SWITCH RELIABILITY,27X,F6.4/9X,15H   FAILURE RATE,29X,F10.4/9X,
    67H   TIME,37X,F10.1/)
 110 FORMAT(9X,42H RELIABILITY FOR FIXED INACTIVE REDUNDANCY//19X,7HSTA
    1NDBY,11X,4HREL.)
 120 FORMAT(19X,I4,10X,E12.6)
 130 FORMAT(1H0,9X,19HELEMENT RELIABILITY,E12.6)
     END
```

```
      SUBROUTINE RELPRB(R)
      DIMENSION X(10),NO(10),TIME(10),FRATE(10),ELCST(10),SPCST(10),
     1SWCST(10),RSCST(10)
      DIMENSION RELSW(10)
      COMMON X,NO,TIME,FRATE,RELSW,ELCST,SPCST,SWCST,RSCST,NEL,TLAM,MS,N
     1,NO
      NN=N-NO
      CALL RELINT(0.0,C0,FCT)
      CALL COE(C1)
      P=0.0
      DO 20 K=0,NN
      AJAY = K+NO
      BINN= BIND = 1.0
      IF( K ) 11, 11, 5
    5 DO 10 I=1,K
      BINN = BINN * FLOAT(NN-I+1)
      BIND = BIND * FLOAT(I)
   10 CONTINUE
   11 C2 = (BINN/BIND)*((-1.0)**K)/AJAY
      IF( MS )12,12,14
   12 P = P +     C2*( 1.0 - EXP(-TLAM *AJAY))
      GO TO 20
   14 CALL RELINT(AJAY,C3,FCT)
      P=P+C2*(C0-EXP(-TLAM*AJAY)*C3)
   20 CONTINUE
      R=1.-P*C1/FCT
      RETURN
      END

      SOURCE LIST
      SUBROUTINE COE(C)
      DIMENSION X(10),NO(10),TIME(10),FRATE(10),ELCST(10),SPCST(10),
     1SWCST(10),RSCST(10)
      DIMENSION RELSW(10)
      COMMON X,NO,TIME,FRATE,RELSW,ELCST,SPCST,SWCST,RSCST,NEL,TLAM,MS,N
     1,NO
      C=1.
      CN = N
      NN = NO-1
      IF( NN )15,15,5
    5 DO 10 I=1,NN
      C = C*FLOAT(I)
      CN = CN*FLOAT(N-I)
   10 CONTINUE
   15 C = CN/C
      RETURN
      END

      SUBROUTINE RELINT(AJAY,        TERM,FCT)
      DIMENSION X(10),NO(10),TIME(10),FRATE(10),ELCST(10),SPCST(10),
     1SWCST(10),RSCST(10)
      DIMENSION RELSW(10)
      COMMON X,NO,TIME,FRATE,RELSW,ELCST,SPCST,SWCST,RSCST,NEL,TLAM,MS,N
     1,NO
      FCT =1.
      K = MS - 1
      IF(AJAY-1.)5,1,5
    1 TERM=(TLAM**(K+1))/FLOAT(K+1)
```

```
      DO 2 I=1,K
    2 FCT = FCT * FLOAT( I )
      RETURN
    5 IF( K )8,6,8
    6 TERM = (EXP((AJAY-1.)*TLAM) - 1.)/(AJAY-1.)
      RETURN
    8 J = 1
      TERM = ((AJAY-1.)*TLAM)**K
      I1=K-1
      I2=0
      I3=-1
      DO 10 I=I1,I2,I3
      J=J+1
      FCT=FCT*FLOAT(I+1)
      TERM=TERM+(FCT*((AJAY-1.)*TLAM)**I)*((-1.)**(MOD(J,2)+1))
   10 CONTINUE
      TERM=(TERM*EXP((AJAY-1.)*TLAM)+((-1.)**(K+1))*FCT)/(AJAY-1.)**MS
   20 RETURN
      END

      SUBROUTINE MODEL(J,R,REL,IN,RB,CB,)
      DIMENSION R(10),IN(10,3)
      DIMENSION X(10),NO(10),TIME(10),FRATE(10),ELCST(10),SPCST(10),
     1SWCST(10),RSCST(10)
      DIMENSION RELSW(10)
      COMMON X,NO,TIME,FRATE,RELSW,ELCST,SPCST,SWCST,RSCST,NEL,TLAM,MS,N
     1,NO
      CB = 0.0
      DO 200 I=1,NEL
      CB=CB+FLOAT(IN(I,1))*ELCST(I)+FLOAT(IN(I,3))*
     1(RSCST(I)+SWCST(I))+FLOAT(IN(I,2))*SPCST(I)
  200 CONTINUE
      DO 100 I=1,NEL
      X(I) = R(I)
  100 CONTINUE
      X(J) = REL
      RB = X(1)*X(2)*X(5)*(1.-X(4)-X(3)+2.*X(3)*X(4))+X(1)*X(3)*X(4)*
     1 (1.-X(5)-X(2))+X(2)*X(4)*(1.-X(3)*X(5)) + X(3)*X(5)
  300 CONTINUE
      RETURN
      END
```